Technical Report 1307

Region-Based Feature Interpretation for Recognizing 3D Models in 2D Images

David T. Clemens

MIT Artificial Intelligence Laboratory

This blank page was inserted to preserve pagination.

Region-Based Feature Interpretation for Recognizing 3D Models in 2D Images

David T. Clemens

This empty page was substituted for a blank page in the original document.

Region-Based Feature Interpretation for Recognizing 3D Models in 2D Images

by

David Taylor Clemens

Submitted to the Department of
Electrical Engineering and Computer Science
in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

Abstract. In model-based vision, features found in a two-dimensional image are matched to three-dimensional model features such that, from some view, the model features appear very much like the image features. The goal is to find the feature matches and rigid model transformations (or *poses*) that produce sufficiently good alignment. Because of variations in the image due to illumination, viewpoint, and neighboring objects, it is virtually impossible to judge individual feature matches independently. Their information must be combined in order to form a rich enough hypothesis to test. However, there are a huge number of possible ways to match sets of model features to sets of image features. All subsets of the image features must be formed, and matched to every possible subset of the model features. Then, within each subset match, all permutations of matches must be considered. Many strategies have been explored to reduce the search and more efficiently find a set of matches that satisfy the constraints imposed by the model's shape. But, in addition to these constraints, there are important *match-independent* constraints derived from general information about the world, the imaging process, and the library of models as a whole. These constraints are less strict than match-dependent shape constraints, but they can be efficiently applied without the combinatorics of matching. In this thesis, I present two specific modules that demonstrate the utility of match-independent constraints. The first is a *region-based grouping* mechanism that drastically reduces the combinatorics of choosing subsets of features. Instead of all subsets, it finds groups of image features that are likely to come from a single object (without hypothesizing which object). Then, in order to address the combinatorics of matching within each subset, the second module, *interpretive matching*, makes explicit hypotheses about occlusion and instabilities in the image features. This module also begins to make matches with the model features, and applies only those match-dependent constraints that are independent of the model pose. Together, the two modules sharply reduce the number of matches that need be subjected to pose-dependent model shape constraints. The modules are demonstrated in the context of a complete recognition system, Reggie.

Thesis Supervisor: W. Eric L. Grimson
Professor of Electrical Engineering and Computer Science

Acknowledgments

I would like to express my sincere appreciation to my advisor, Eric Grimson, for his many precious hours spent discussing and reviewing ideas, and for his financial support. His advice has been invaluable, and his dedication has been inspirational. I would also like to thank committee members Ellen Hildreth and Berthold Horn for their careful readings and helpful comments.

During my years at the MIT Artificial Intelligence Laboratory, I have found my peers to be a vital force, both intellectually and emotionally. In particular, my friendship with David Jacobs and our many fruitful discussions have been very important to me, and I hope they will continue for many years to come. Insights from many other people, Jim Mahoney, Todd Cass, Sandy Wells, Shimon Ullman, Tomas Lozano-Pérez, Eric Saund, Dan Huttenlocher, Dave Braunegg, Ronen Basri, Amnon Sha'ashua, Yael Moses, Tao Alter, Brian Subirana and many others were also greatly appreciated, and have shaped my work considerably.

The lab environment is all too easy to take for granted, and I would like to thank all the people who have contributed to its welfare, though I may never really know who they are or how much they have given. This includes all those who put their energy into funding, inspiring, administering, debugging, facilitating, and supporting the everyday operations that make the lab a working reality.

For their support and friendship, I would like to thank my parents, my brother, and, in addition to those above, Karen Sarachik, Jonathan Amsterdam, Nomi Harris, Jeanne Speckman, Liz Edlind, Barb Moore, Jeff Koechling, Nancy Pollard, Mike Kashket, Leif LaWhite, Scott Huler, and Eve Berne, who stood by me through the hardest times. I feel very lucky to have known such remarkable people, and I look forward to many more adventures with them.

This document describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by an Office of Naval Research University Research Initiative grant under contract N00014-86-K-0685, and in part by the Advanced Projects Agency of the Department of Defense under Army contract number DACA76-85-C-0010 and under Office of Naval Research contract N00014-85-K-0124.

Contents

Chapter 1

Introduction

1.1 The Vision Problem

If we hope to build machines that are flexible and autonomous in the human world, we must address the need for some kind of sophisticated sensory mechanism. We know what solutions evolution has found– almost every creature can see, hear, touch, and smell. Of the senses, the most powerful and complex is sight. By projecting reflected ambient light in an orderly manner onto a dense array of sensors, humans are able to perceive the spatial nature of the world around them at a glance, and with great confidence. Though we have developed alternative sensing technologies, the biological example of sight remains the most compelling for performing human-like tasks in a variety of environments.

Vision, however, is very difficult to understand and recreate. Consider the visual task of locating and recognizing objects in the three-dimensional (3D) world from projected two-dimensional (2D) images. In performing this task, the essential difficulty is that the 2D projection of a particular object in an image can be so varied. First, the image of a typical object changes greatly from different viewpoints. An object may have rigid shape in 3D space, but when it is projected into a 2D image it may take on various appearances that are not related by rigid 2D transformations. Part of the 3D shape information is lost, and the rest is confounded. Second, the parts of the image due to a particular object are generally difficult to distinguish from the rest of the image. Attempts to segment the image directly into separate objects have been quite unsuccessful to date. Third, the appearance of an object will depend on the entirely unpredictable state of the rest of the world, in a variety of ways. Neighboring objects may partially cover (occlude) the object. The intensities in the image will depend on lighting, viewpoint, reflections, and shadows from other objects. Further, for an object that is not rigid, there are a whole family of shapes associated with it. It is very difficult to characterize this family while still retaining whatever essential

shape information distinguishes it from other objects. All together, if it were not for the fact that biological vision systems exist, it would be hard to believe that the task is possible at all.

Because vision is so difficult, it has been necessary to break the problem into parts or domains, and focus research within each domain separately. There are two dangers to dividing the problem. First, since the goals within each domain are more limited, solutions discovered within the domain may not generalize to a larger, more practical domain. Second, since the sources of information are usually limited in each domain, each subproblem is likely to be somewhat harder. There is no guarantee whatsoever that the restricted domain corresponds to a nicely encapsulated module in natural vision systems. In choosing a domain, these dangers should be heeded, even if they cannot be completely avoided.

In this thesis, I address the problem of finding the position and orientation (or *pose*) of a modeled 3D object in a single 2D image of the natural indoor world (Figure 1.1). The modeled object must be partly-polyhedral, and should not have much visual texture (tiny markings or bumps). The image is an array of grey-levels, formed under perspective projection. While humans certainly gain much perceptual information from motion, color, and stereo, they are also able to do without these sources and successfully interpret black-and-white photographs, with almost as much ease and only slightly less accuracy than if they were viewing the scene directly. The task should therefore be possible. The restrictions of an indoor world and semi-polyhedral models are derived from the need for repeatable features on the objects, while the absence of visual texture allows regions to be found in the image. However, the domain is rich enough that we may reasonably hope to extend the results to more general vision problems. Despite the restrictions, the problems within this domain continue to prove challenging—they are on the forefront of model-based recognition.

Within this domain, a common approach has been to find local features in the image, and to match them to features in the model. In this way, the image "signal", which can vary so greatly, is abstracted into symbols that are interpreted at a higher level. The problem is thus divided into two parts: finding stable image features, and matching them to the model. The desired output of a feature-based vision system is a correspondence between the visible model features and their counterparts in the image, and the pose of the model (relative to the camera) such that the model would appear as it does in the image (Figure 1.2).

Both feature-finding and matching have been important issues in vision research. In this dissertation I focus on the matching problem. This presentation of the matching problem makes several assumptions, such as the atomic nature of features, and the one-to-one correspondence between visible model features and correct image features, but it captures difficulties of the problem that persist in more general formalizations. In finding matches between the image features and the model features, the following

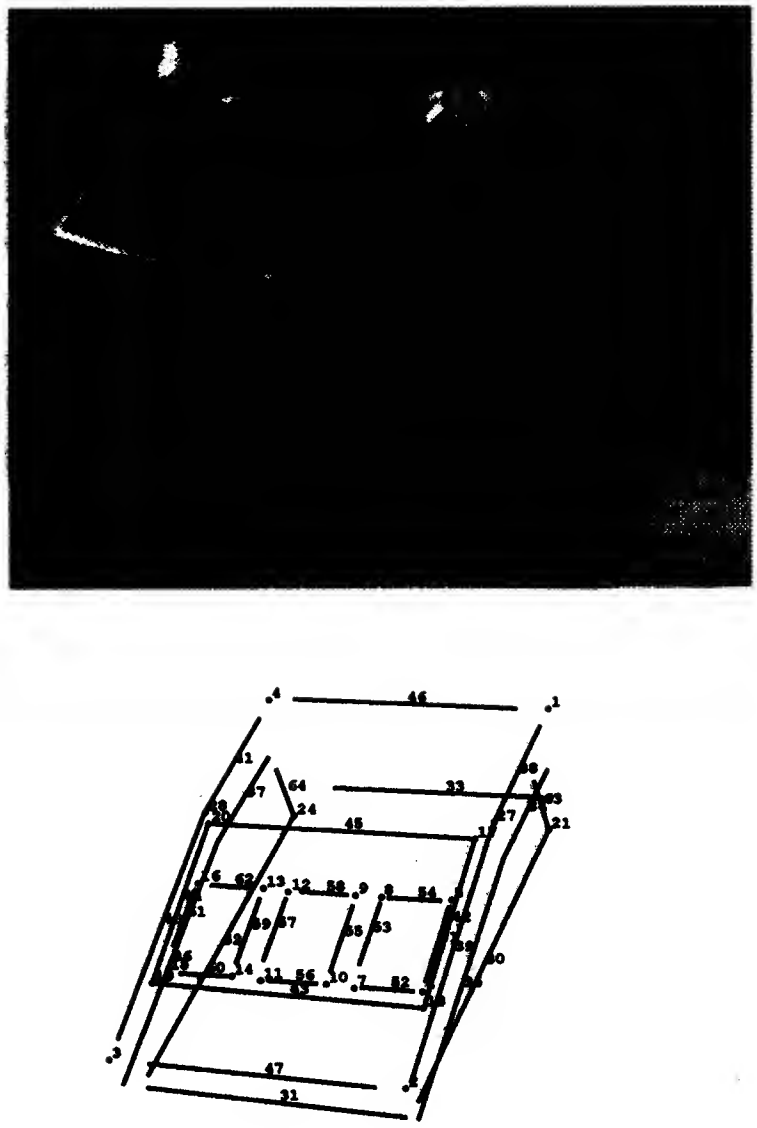

Figure 1.1: *A typical image and model, which serve as inputs for the recognition task.*

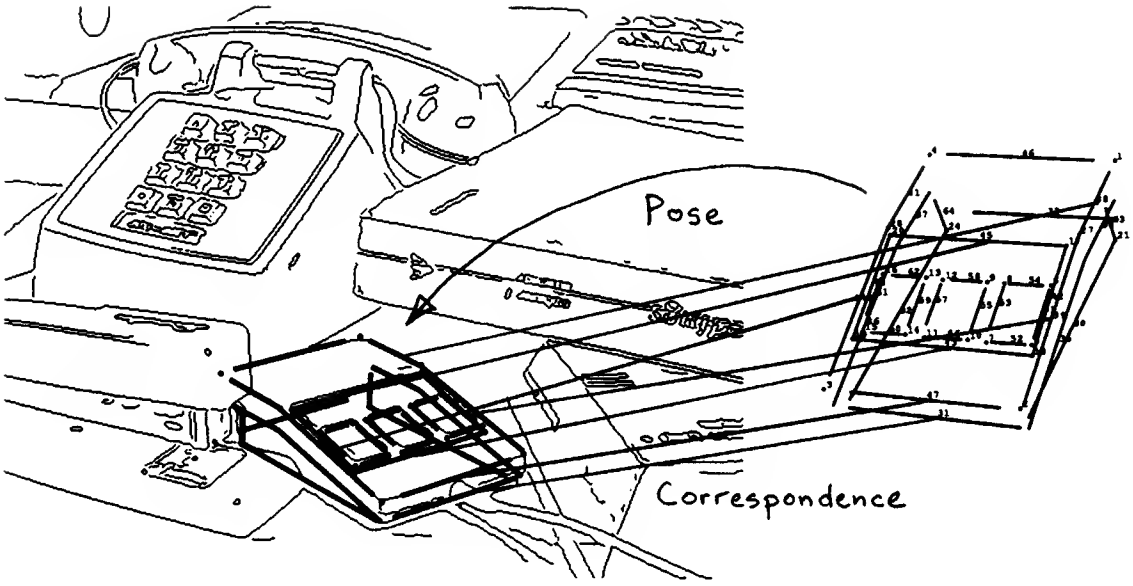

Figure 1.2: *The desired output of recognition is the pose of the model that aligns it with the image, and the correspondence between features in the model and image, for each instance of the model in the image.*

phenomena are common and must be addressed:

1. Many image features do not correspond to any model feature.
   (a) Many image features are from other objects in the image.
   (b) Many image features are caused by shadows, highlights, and other unpredictable events.
   (c) Extra features may be found on the image of the object that are not included in the model. The features may be omitted from the model because they appear too rarely.
2. Some model features do not correspond to any image feature.
   (a) From a particular viewing direction, some model features will not be visible.
   (b) Other objects may hide (occlude) parts of the model object.
   (c) Model features that should be visible in the image are often not detected, or may be distorted. The distortion may be inherent in the gray-level image or due to the feature extraction process.

3. Image features are (probably) not elaborate enough to be identified with model features by their own properties alone.

   (a) Features must be small enough to avoid being partially hidden.

   (b) Features must be simple enough to be reliably detected from a wide range of viewing directions.

One way to account for all of these problems is to form all combinations of matches. First, form all subsets of the image features (to account for 1), and all subsets of the model features (to account for 2). Then form all pairs of image and model subsets of equal number. Finally, match the features within each subset pair, in all permutations (assuming 3). Then, for each match combination, solve for the best model pose and test the accuracy of the alignment. This will produce the best match set, by definition, but it is an impossibly large search to perform. If there are N features in the image, and M features in a model, then there are

$$\sum_{G=1}^{\min(N,M)} \binom{N}{G}\binom{M}{G} G!$$

possible complete correspondence hypotheses. (This is bounded above by $(N+1)^M \approx N^M$, which is the number of ways to assign each model feature to one image feature or to a null match, allowing multiple assignments to the same image feature and assuming $N > M$.) Suppose there are 100 image features and 10 model features, and the pose solution and alignment test takes only one one-millionth of a second per match (the current implementation actually takes about one full second). Then this approach would take over three million years to execute.

However, the search does not have to be conducted in a blind generate-and-test method. There are various ways to constrain the search, and thereby generate only those matches with greater potential for being acceptable. The great majority of the impossible matches can be eliminated without being individually considered, as long as the phenomena listed above are addressed. The goal of much model-based vision research, and of this thesis, has been to explore ways of constraining the match search.

1.2 The Thesis

Theoretically, it is impossible to uniquely determine a 3D structure from a 2D projection of it. There are an infinite number of 3D worlds that will cause the same 2D projection. The only reason that we can hope to accomplish this visual task is that

we know things about the world—we have further constraints. But the constraints are complex and very difficult to characterize. The essential task of vision research is to understand these constraints, and to apply them efficiently.

In model-based vision, the most direct constraint is the shape of the model. If a particular set of image features are indeed from an instance of the model in the world, then they must be geometrically consistent with some view of the model. Since the degree of geometric consistency is usually the final criterion for accepting or rejecting a hypothesis, it is a natural candidate for constraining the search at earlier stages. One common approach is to make all permutations of match sets with a small fixed number of matches. In each match set, there must be just enough matches to determine a model pose with sufficient accuracy to predict the best completion of the match set. But the combinatorics of matching even small numbers of features have been a difficulty. Another general approach has been to derive aspects of the model that are pose-invariant, and search for these directly instead of finding the pose. Still, to provide enough constraint, this involves collecting matches in many combinations. The various solutions have certainly been improvements on the $O(N^M)$ approach, but more constraint is needed.

Fortunately, there is an another important class of constraints that can be brought to bear: *match-independent* constraints. These come from general knowledge about the kinds of objects in the world, the behavior of light, and the image formation process. By using this information, the image may be *interpreted* to some degree before making any matches to specific models.

The process of interpreting the image is not new by any means—it was the way machine vision research started. Before the recognition task was addressed, machine vision researchers tried to "understand" the entire image without reference to modeled objects. [Waltz 75] attempted to identify the causes of edges in the image, and to deduce which edges came from the same object. [Marr 77] proposed that the depth at each pixel in the image should be found, and [Barrow and Tenenbaum 81] proposed that interior shadows, cast shadows, and other image properties be classified at each pixel. But to find unique global values for these properties is a very difficult task, and the research met with limited success. Instead, similar analyses can be performed in the context of the recognition task. In support of recognition, the analysis may be local, may include multiple and inconsistent interpretations, and need not be performed at every point in the image. Under these conditions, match-independent analysis can be very successful. This insight is the general motivation for this thesis.

The approach is inspired primarily by recent work in machine vision by [Lowe 85]. Lowe describes a recognition system that incorporates some of the concepts of Perceptual Organization, borrowed from the field of psychology. Perceptual organization is the general process of "making sense" of the image before identifying objects, as revived by [Witkin and Tenenbaum 83] and others, and dating back to [Wertheimer 23].

Match-independent deductions about the viewed world can be used to greatly reduce the complexity of the matching problem. Specifically, I focus on two goals for image interpretation: to determine to some extent (1) which features belong to the same objects, and (2) which features are likely to be stable. Both of these goals involve estimating the spatial relations among objects (locally), especially whether one object is in front of another. To the extent that these facts can be hypothetically determined, many fewer combinations of matches are required. If groups of image features that come from a single object can be found, only those groups need be considered, since it is impossible for image features that come from multiple world objects to match the same model object. This alleviates the need for forming all subsets of image features, while still addressing the existence of image features from other objects (1a). Likewise, when an object is known to be partly covered by another in the image, there is no need to find matches for the parts of the model that are hidden. This explicitly addresses 2b, avoiding the need to test all combinations of hidden model features. Similarly, when image features display signs of being unstable (1c), only they need to be tried in multiple combinations, rather than all image features. The performance and reliability of the interpretation process are important issues that remain to be justified, but its potential to address the matching problem head-on is very attractive. In general, I believe the concept of employing match-independent knowledge is essential to the development of practical machine vision systems.

Towards that end, I have developed two modules, *region-based grouping* and *interpretive matching*. Figure 1.3 indicates how these modules are organized in a complete recognition system, and how other modules also support the application of match-independent constraints. (It is not a system diagram—see Chapter 3 for a detailed explanation.) While the detailed approach of each module may be of practical interest, the primary purpose of creating the system is to demonstrate the value of the general strategy. Chapter 6 contains the complete results of experiments, and an analysis. Typically, out of billions of possible minimal matches per image, the system generated only several hundred for pose-dependent verification and still succeeded in recognizing the modeled object in most cases.

The first module, Region-Based Grouping, finds open areas of the image where there are no intensity edges. The edge-based features surrounding the open region are then grouped together. The motivation for this approach to grouping is simply that objects tend to have smooth surfaces surrounded by high curvature ridges. When projected, the high curvature ridges often cause spatially abrupt intensity changes (edges), while the surfaces often do not. This may not be true for all objects under all circumstances, but it is a sufficiently common phenomenon that region-based groups are very useful.

The second module, Interpretive Matching, combines two very different kinds of

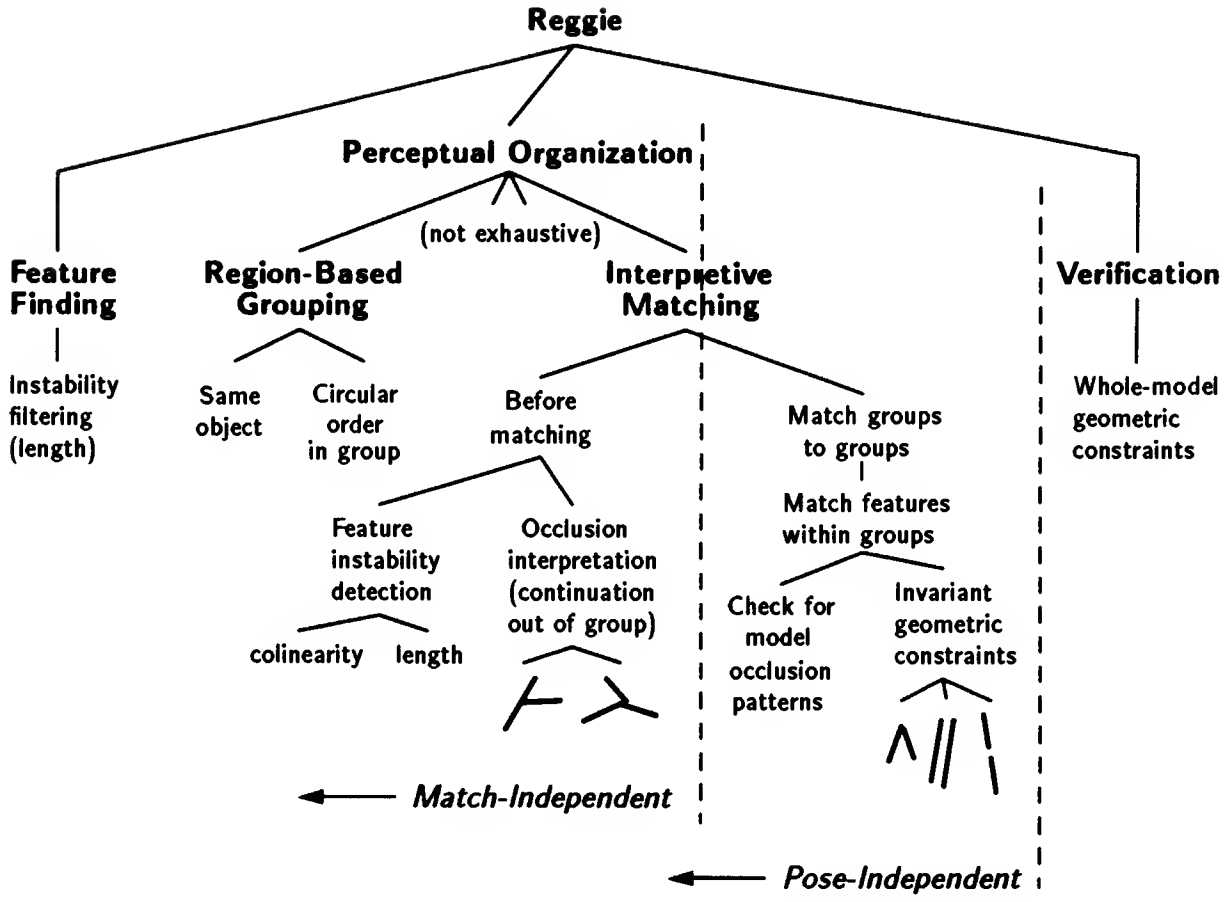

Figure 1.3: *Module components and objectives pertaining to interpretive matching. (This is not a system diagram).*

constraint. First, it uses match-independent image interpretation to form hypotheses about occluded and unstable features. Second, it forms matches between the image and model features, and applies match-dependent (but pose-independent) constraints. It starts with one model group and one image group, and must generate feasible correspondences between the features within the two groups.

The region-based groups of the first module are a particularly appropriate starting point for the occlusion analysis. There are a relatively large number of features in each group, which allows a somewhat more sophisticated interpretation, and a larger collection of constraints. In addition, the order of edge-based features around the region is preserved under projection. This significantly restricts the possible permutations in matching to the model features. Most basically, the concept behind region groups plays an important role in hypothesizing occlusion, so that specific image features can be omitted from the match. If the open region in the image is in fact from a recognizable part of a single object, then the boundary features tend to form a (semi-) continuous perimeter. Occluding objects cause invasions into this perimeter from outside. The detection of an invasion relies to some extent on the continuity of the object boundary, the prevalence of features from the object as reference, and the local definition of inside and outside, all of which are supported by region groups.

In addition to these two modules, an important contribution of this thesis is the construction of a complete vision system, called Reggie (for REGion-based Grouping and Interpretation of Edges). With a complete system, the utility of the modules can be measured by their successful contribution to the entire task, rather than by a subjective visual inspection of their output.

A complete system also allows the *robustness* of its parts to be tested. Few vision researchers explore this important dimension, partly because it is more nebulous than other criteria. The domain of successful applications is difficult to determine and describe. It is relatively easy to judge whether the result of a recognition system, the pose and feature correspondence, is correct. As well, the execution time and space requirements of a system are two important parameters that are easy to measure. But, despite the accessibility of these quantities, the overall success of a vision system is quite hard to ascertain. The difficult question is, under what conditions will the object be reliably found? For example, can it be fifty percent hidden? Can it appear at any size in the image? Will a shadow across the object prevent discovery? What kinds of objects can be successfully modeled and recognized? What kinds of neighboring objects can be tolerated in the image? Model-based vision systems to date have been successful only in narrow ranges of possibilities, and the ranges have not been well defined. I report on experiments to answer at least some of these questions about Reggie.

In the rest of this chapter, I describe in greater detail how the Reggie system

relates to the field of machine vision, and how it addresses the issues of efficiency, accuracy, completeness, and multiple model instances. In the two following chapters, I present the region-based grouping module and the interpretive matching module. In each chapter I discuss related work. In Chapter 4 the rest of the Reggie recognition system is presented. Then, in Chapter 5, I describe a separate component of Reggie that is used to create three-dimensional object models from two-dimensional images. Finally, in Chapter 6, I present the experimental results, and offer conclusions in Chapter 7.

1.3 Aspects of the Recognition Problem

In this section, I introduce the issues of completeness, accuracy, multiple model instances, and efficiency, which are the topics of the remaining sections in this chapter.

The goal of model-based recognition is to find instances of modeled objects in an image. By "instances", what is presumably desired is "instances that a human can find". But the abilities of the human visual system are very sophisticated, hard to quantify, and not formally understood. Even an apparently quantitative goal, to find all instances that are more than 30 percent visible, cannot formally be achieved because it still makes reference to human perception: which features are "visible"? All human observers might be able to agree on exactly which parts of the image correspond to the model, but the definition cannot be expressed formally, because human perception is not understood formally. In practice, some scoring function must be defined, explicitly or implicitly, that determines the correctness of a hypothesis in terms of the image and models alone. It will not have access to the human answer. The scoring function may attempt to evaluate the percentage of the model that is visible, for example, by totaling the lengths of the model edges that have been matched in the image. But because there is no way for the system to know which matches are correct, there is no way to assure that the scoring function is behaving in accord with human perceptions. In fact, there is often a significant difference, and this is one of the main reasons that recognition systems are not robust.

Therefore, it is useful to separate the justification of a vision system into two parts: an argument for the likelihood that the scoring function is reasonable with respect to the desired task, and a guarantee to find all hypotheses with the desired scores. The argument for the scoring function is important, but simply cannot be formal when success at the task is defined in terms of human performance. However, given the scoring function, the guarantee to find all desired hypotheses is within the realm of formality. It is often called *completeness*, since it implies that the search for high scores has effectively considered every candidate. If completeness can be guaranteed, then the accuracy of the system is guaranteed to be as good as the scoring function.

Completeness is the subject of Section 1.5.1.

The scoring function used in Reggie is not unusual, and is described in Section 4.2. However, in addition to the definition of the scoring function, it is important to define how it will be used. If the task is to find a single instance of a modeled object in an image, then a "best-first" search may be combined with *early termination* to increase the expected efficiency. With early termination, the first hypothesis that scores above a threshold is accepted, and the search is terminated. (Since ordering the search does not guarantee that later hypotheses will lead to lower scores, early termination compromises completeness.) But, for most vision tasks, a single instance of the model cannot be assumed; there may be one or more instances, or no instances, and each of these cases should be detected. This is particularly important when a library of models is to be accessed, since many of the models may not appear in a given image. The detection of all instances is a much harder task, since the search must exhaust all hypotheses with any chance of scoring above a threshold, precluding any advantage of best-first search. It also requires a scoring function that rates all correct hypotheses above a threshold and all incorrect ones below, rather than a score that simply orders the hypotheses relative to one another. The Reggie system has a search structure that supports the discovery of all model instances, as discussed in Section 1.5.2.

While the definition and accuracy of a vision system are of primary concern, ultimately its efficiency also determines its success. The system must be practical in some sense. Whether the cutoff is an hour or a year, it cannot take too long to execute, or require too much space. This issue is very implementation-dependent, but it is currently an important driving force in vision research. It rules out the approach of storing all possible images, for example, because the variations (viewpoint, illumination, neighboring objects) are far too numerous. As a less extreme example, as presented earlier, the need for efficient execution has ruled out the possibility of explicitly forming and testing all combinations of complete matches between the image features and the model features. A successful recognition scheme must be much more efficient than this. However, there is a trade-off between efficiency and accuracy. To reduce the combinatorics to acceptable levels while still preserving sufficient accuracy is the goal of grouping and interpretation. These are the subjects of the next section.

1.4 Defusing the Combinatoric Explosion

Let a single correspondence between an image feature and a model feature be called a *match*, and let a set of matches constituting a hypothetical identification of the model be called a *match set*. As presented above, N^M possible match sets are far too many to evaluate explicitly. In the Reggie system, three important concepts are applied to

reduce drastically the number of explicitly explored hypothetical matches. In order of execution they are: grouping, interpretation, and alignment. While grouping and interpretation are the focus of this dissertation and have been introduced above, alignment is a common recognition approach that sets the stage for their combinatorics, and so is discussed first.

1.4.1 The Combinatorics of Alignment

The score of a hypothesis is based on the ability to make the model features appear like the corresponding image features. The transformation from model to image involves rotating and translating the model features as prescribed by the pose, and projecting them into the image plane. The differences between the positions of the image features and the projected model features are then combined to determine the error. The error will therefore depend not only on the hypothesized matches, but also on the pose of the model.

If all possible match sets were hypothesized, they would then each have to be evaluated in terms of the scoring function. In that case, the correspondence is declared in the hypothesis, but the pose is not. The pose that minimizes the error is desired, and the minimized error is used as the score to judge the match hypothesis.

The problem with this scheme is the huge number of complete match sets, and the cost of verifying each one. Instead of just trying all of them, we would like to find a way to direct the search towards those that will produce a better score. If the pose is known, for example, then the model features can be projected into the image and simply matched to the closest image features. But the space of all poses is also huge, since the pose has at least six degrees of freedom. It is therefore generally prohibitive to sample the pose space and explore it completely. (Many approaches do use the pose space to organize information, however[Ballard 81] [Besl and Jain 85] [Thompson and Mundy 87] [Baird 84] [Cass90]).

Alignment is an intermediate solution between pure correspondence and pure pose search. If a small number of matches are hypothesized (forming a *partial match set*), then the pose can be found which aligns these features with minimum error. For example, if three image points are matched to three model points, two poses can be found analytically such that the error is zero ([Huttenlocher 88]). Then, using those poses, the rest of the visible model features can be projected into the image, and the best nearby image features can be found relatively efficiently to complete the correspondence. This method was described in [Fischler and Bolles 81], and versions of it can be found in [Lowe 87] and many other recognition systems. It has recently been reemphasized and named Alignment in [Ullman 86]. In this method, the small initial match set is considered to be the hypothesis, and the processes of finding more matches from the model projection and determining the error are together called

"verification".

If all permutations of three matches are formed, there are only $O(N^3M^3)$ hypotheses. For each hypothesis, the verification stage is $O(NM)$, since each model feature will be compared with each image feature to find the closest match. (This assumes that under these conditions, the best match for each model feature can be found independently.) Therefore, using alignment, the search for the best match has been reduced drastically from $O(N^M)$ to $O(N^4M^4)$. The question is, might some desired match sets be missed? A careful answer to this question is a matter for further research. Essentially, it is possible to miss the desired match sets, but the probability may be acceptable. The possibility arises because the initial pose, found by minimizing the error of just three seed feature matches, will probably be different than the pose that minimizes the error of the whole match set. Thus the initial pose will not necessarily align the rest of the model features with the desired image features. A small orientation error around the seed matches will cause other model features to have a displacement proportional to the distance from the seed match (see Figure 1.4). Incorrect image features may then be closer than those in the desired match.

Even if the pose is updated as each new verifying correspondence is made, (as [Ayache and Faugeras 86] have done in the domain of 2D data and 2D models), there is still no guarantee that the desired match set will be found. Since the pose is initially based on three matches, and the current pose determines which new matches are made, a false match is possible and will lead the pose away from later correct matches.

There is an important reason why the desired match set is likely to be found, however. Of the many possible subset triples of the desired match set, if *any* work, that is good enough—it only takes one. Nonetheless, if larger initial hypotheses could be formed, it would improve the accuracy of seed poses. The seed pose for four matches (for example) will more closely approximate the best pose for the correct match set, and so will be more likely to generate the correct correspondence.

However, even $O(N^4M^4)$ is too large. As shown in Table 1.5, the execution time is beyond the reasonable range, under the given typical conditions. This is particularly clear for partial match sets of size four, which require $O(N^5M^5)$ search times, if all combinations of seed matches are to be tried. Instead, some kind of grouping must be performed.

1.4.2 Grouping Combinatorics

Grouping provides useful seed matches without generating all possible combinations. There are many kinds of grouping, but the region-based grouping method presented in this thesis is argued to produce a number of groups that is proportional to the number of features. (It accomplishes this in linear time, which is not significant compared to

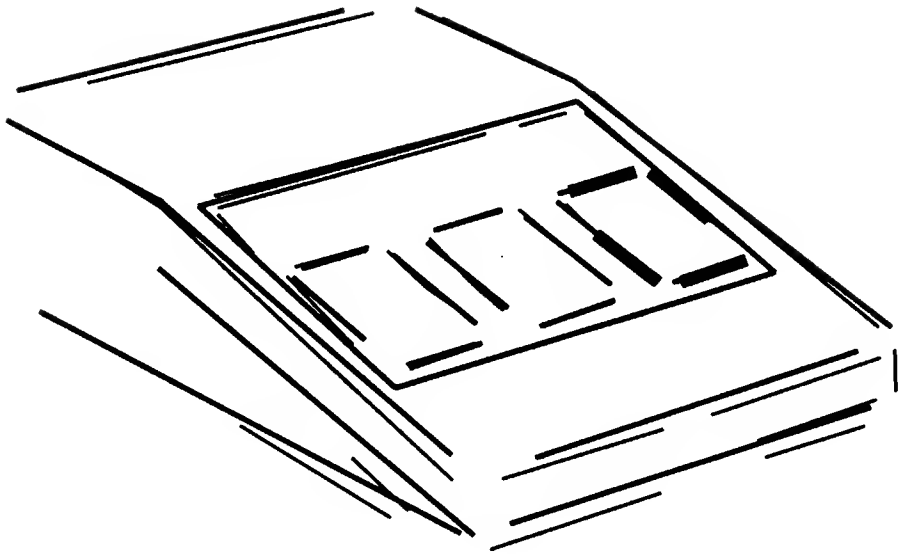

Figure 1.4: *The pose of the model was determined from the four seed matches in the right button. The rest of the model features (bold) were then projected on top of the image features. Due to small errors in the pose, the other model features may not align with the correct image features. The deviation tends to be larger if they are further from the seed matches.*

Matching Strategy	Number of Match Sets	$N = 100$ $M = 10$ $G = 7$	At 1E-6 second per match set
All combinations	N^M	1E20	3 million years
Alignment, G matches	$(N^G M^G / G!)NM$	2E17	6300 years
Alignment, 4 matches	$(N^4 M^4 / 4!)NM$	4E13	1.3 years
Alignment, 3 matches	$(N^3 M^3 / 3!)NM$	2E11	2 days
Align + Region Groups	$(NMG3^G)NM$	2E10	4 hours

Figure 1.5: The combinatorics of several matching strategies. The time values are only included to give a feeling for the numbers—they are not real system execution times, and they completely ignore constant factors, match generation times, and typical-case performance.

the time of verification.) This is discussed in Chapter 2.

Since grouping must provide seed groups to the matching module, it does not depend on any matches. It can therefore be applied to the image features independent of the model features. To further reduce the combinatorics, corresponding grouping operations can also be applied to the model features, independent of the image. Together, these operations bring the number of group matches down to $O(NM)$, given that the grouping method is linear-time. For each group match, the correspondence between the features in the image group and the features in the model group must be made, allowing for all possible unmatched features. If all combinations were possible, there would be

$$\sum_{i=1}^{G} \binom{G}{i} \binom{G}{i} i!,$$

or approximately G^G possible combinations, where G is the number of features in a group. (G may vary among groups, but the largest group dominates the asymptotic execution time. The approximation follows the same reasoning as the N^M approximation.) However, in addition to restricting the number of groups, region grouping also preserves the circular order of the features in each group. The only requirement is that the object be opaque, and that enough features bound the region to define where the inside of the region is. With circular order preserved, there are

$$\sum_{i=1}^{G} \binom{G}{i} \binom{G}{i} i$$

possible combinations. This is bounded by $G3^{2G-1}$ (as can be seen by drawing a search tree, as in Chapter 3, Figure 3.1b), and can be approximated by $G3^G$ for small G. The overall match-and-verify combinatorics are thus reduced to $O(N^2M^2G3^G)$, as shown in the table.

Efficiency is greatly increased, but at what cost to accuracy? If grouping is only performed in the image, and not in the model, then the correct match set will be discovered if at least one group contains a sufficient number of correct matches to the model features. Even a sparse proximity grouping technique could satisfy this criterion with high probability. However, when groups are formed in both the image and model, and feature matches are allowed only within group matches, the criterion is more strict. The correct match set will be discovered if, at the minimum, one of the image groups contains a large enough number of correct matches to features from one of the model groups. That is, there must be at least one sufficiently large intersection between the image groups and the model groups, as visible in the image. The minimum number of correct matches depends on how well alignment works, so it might be three, four, or possibly more. The accuracy therefore depends on the repeatability

of the groups, which is very difficult to determine analytically. Empirically, however, the region groups formed by Reggie are well above this minimum requirement.

1.4.3 Interpretive Matching Combinatorics

Even with grouping, all combinations of image and model features are formed within each group match, preserving circular order, but including all combinations of omissions. The formation of all combinations accounts for occlusion, feature detection instability, and grouping instability, but it creates $G3^G$ match sets for each group match, which is exponential in G. The Interpretive Matching module uses both knowledge about the world and about the model to anticipate occlusion and unstable features, and to form multiple interpretations only when necessary. The details of this operation are described in Chapter 3.

The interpretive matching operation replaces the last portion of "all-combinations matching". But its combinatorics are difficult to analyze, and so is the resulting accuracy. The matching process begins by trying every one of the G initial matches. From each starting match, it proceeds around the image group and the model group in step, accruing matches as it goes. If conditions appear in the image that indicate an occlusion, then the image features from the occluding object are skipped. After an occlusion, since it is unknown how many model features were hidden, all possible numbers of consecutive model features are skipped. This causes branching in the search, which is executed depth-first. In addition, in case the occlusion hypothesis is wrong, the direct match is continued in another branch. Branches also occur when there is evidence that two image features come from one model feature, or when an image feature is very short, or when a model feature is marked as a likely generator of unstable image features. If at the leaf of a branch there are extra image features or extra model features, the match is abandoned.

Even with this many branching situations, the interpretation will create fewer than all combinations of match sets. At each step around the groups, there are three possible branches: match the image feature to the model feature, skip the image feature, or skip the model feature. Without interpretation, all three branches are explored (and thus the three in $G3^G$). With interpretation, many of these branches are eliminated, and no additional branches are created.

To further reduce the number of branches explored (and further confound the analysis), pose-invariant constraints are imposed after each match. If two lines are parallel in the 3D model, then the corresponding image lines must also be nearly parallel, assuming perspective "distortions" are not significant. Likewise, if two model lines are colinear or are connected at their endpoints, they must do the same in the image (again, within some bounds, to account for perspective and variation in the feature extraction process). If any constraint is violated, the match set is abandoned.

Exactly how many match sets will result from this approach, and how many nodes will be explored in the process, is hard to predict. Also, whether the correct match will survive the interpretation depends on the ability to detect occlusion and other image phenomena. In the absence of analysis, the interpretive matching module was implemented in order to explore empirically the efficiency and accuracy of some specific approaches. The results are presented in Chapter 6.

1.5 Completeness and Multiple Model Instances

1.5.1 Completeness

As introduced above, completeness is the ability to guarantee that a recognition strategy will find *all* hypotheses with the desired scores (such as the highest score, or all scores above a threshold). Due to the huge number of possible match sets, most systems must eliminate large portions of the search space without evaluating each member. To eliminate a whole collection of match sets at once, they must have some property in common. In order to prove completeness, it must be shown that absolutely none of the eliminated match sets could have achieved the desired score, based only on the common property.

If the search is structured as a tree, conservative pruning can be demonstrated to preserve completeness. In this case, all the match sets growing from a particular node of the tree have in common all the matches up to that node. To support early exclusion, the scoring function may be structured such that partial match sets can be given partial scores, and an upper bound on the complete score can be found. For example, the score may be defined as the product of the "closeness" of each projected model feature to its corresponding image feature (or image features), where closeness is measured between zero and one. The pose that maximizes the score is used. If a partial match set has a partial score that is below the acceptable threshold, then every match set that includes that partial set may be safely excluded from consideration, since additional matches could never increase the score.

This kind of pruning is only available for branches in which the partial score is already unacceptable. More drastic pruning is usually needed. In the alignment approach, a partial match set is used to find the best pose, project the model into the image, and complete the match set. In using the hypothetical pose to find new matches, a very large set of matches are excluded from the search. For each model feature, only the closest image feature is typically considered, and the best match is judged independently of other model matches. In order for this method to be complete, it must be argued that *any one* of the excluded matches would have brought the score for the entire match set below the desired threshold. Further, the argu-

ment must hold even allowing for the fact that a better pose could be found for the larger match set. In other words, it is insufficient to show that an image feature is too far from the current projection of a model feature, based on the best pose for the established matches. It must also be shown that any pose that brings the model feature close enough to the image feature would cause excessive error in the established matches. If this test were implemented, the large number of match sets would probably make the process unattractive. In general, the alignment method of search is not complete. As argued above, however, it can be reasonably accurate if the hypothetical pose is good enough.

A much more direct violation of completeness is early termination. In this approach, the search through the possible partial match sets is ordered by some criterion, in an attempt to investigate the best first. Each partial match set is explored in turn, and as soon as one achieves a sufficiently good score it is taken as the result, terminating the search. The completeness of this approach hinges on the ability to find the best complete match set before finding any other sufficiently good one. Clearly, this cannot be guaranteed. If the partial scores are a certain indication of the complete scores, then the scoring function must be ignoring the completion of the match set. (Alternatively, if the result of the search can be guaranteed before it is completed, then the discarded portion was never a feasible part of the search. It is perfectly acceptable to ignore branches based on what has already been discovered, but the test must be performed for the branch before it is ignored.)

From these examples, we can see that completeness is always relative to the scoring function; it cannot be absolute. Put another way, completeness is only as desirable as the scoring function. In particular, the scoring function is often expressed in terms of the image and model features, not directly in terms of the image grey-levels. For example, it may explicitly measure the proximity of image features to the projected model features, and it can be argued that this makes intuitive sense. Implicitly, however, this is based on assumptions about the choice and derivation of the features and edges. The features are a "given" part of the scoring function: given the image and model features, the closest match set is found. In arguing for the relevance of the scoring function, the implicit assumptions must also be defended.

Completeness can always be guaranteed relative to *some* scoring function. Simply define the entire recognition algorithm as the "given" part of the scoring function, and completeness is trivially assured. The burden now rests on justifying the entire system as a reasonable scoring function. This example is extreme, but it demonstrates the tradeoff that exists between formal and informal arguments. The tradeoff puts the completeness criterion in perspective—it is not as absolute as it sounds.

In fact, some flexibility in the definition may be in order when evaluating the completeness of grouping. If the scoring function is expressed in terms of feature error, grouping is not complete. There is some chance that all of the possible groups

leading to the desired match sets will be excluded. But it can be argued that grouping should be considered an implicit part of the scoring function, as is feature extraction. The general knowledge invoked by grouping certainly increases efficiency, but it is also likely to increase accuracy[Jacobs 89]. It therefore might be considered a worthy implicit contribution to the evaluation of the match set. And yet, to accept groups as "given" in the scoring function would dismiss their inaccuracies. Perhaps, instead, the inaccuracies of feature extraction should be considered part of the matching problem, rendering all approaches incomplete. Primarily, this emphasizes the relative nature of the completeness criterion.

Reggie uses alignment, grouping, and interpretation, none of which are complete. However, it does *not* use early termination, and the importance of this is discussed in the next section.

1.5.2 Detecting Multiple Model Instances

One desirable property of a recognition system is that it degrade gracefully as the visual task becomes harder. When less of the model is visible, for example, the search might take a little longer, but not an unacceptable amount of time. The probability of a false answer might go up steadily, but not abruptly. With this goal in mind, it is often thought that using thresholds to terminate search is not attractive. This is one motivation behind the best-first search: rather than discarding all partial matches below some partial score threshold, order the partial match sets by their partial score, and complete them in order until a sufficiently good match is found, then terminate. In this way, it will take longer to find the best hypothesis when it is less distinctive, but the behavior will not change drastically if the best score drops below some threshold.

However, this approach simply cannot be used if the image contains either multiple instances or zero instances of the model, and if the detection of these cases is part of the vision task. If multiple instances are to be found, and the number of instances is unknown, then early search termination is not appropriate since it produces only one instance per invocation. If it is invoked until it finds no further instances, then it is not early termination. But even without early termination, a best-first search will have to apply a threshold at some point. If there are no instances of the model in the image, the search will continue exploring beyond the "best" partial match sets until it reaches some lower bound of goodness. The search space is too large to explore in its entirety. If there are multiple instances, the search cannot use the instances already found to affect the termination; again, it must use a threshold. In order to handle the no-instance and multiple-instance cases, a threshold on the scoring function must be used to terminate the search.

In the Reggie system, all sufficiently good match hypotheses are verified. This might be called an "all-good" search. Throughout the system, conservative cutoffs

are made at every stage, and all hypotheses that pass the cutoff are always considered further. This means that, relative to the scoring function, all sufficiently good instances of the model in the image are found, in an amount of time that does not depend significantly on the number of instances.

When an all-good search can be performed efficiently, it is more likely to be accurate. It is easier to confidently determine whether a partial hypothesis has some chance of being correct than it is to assert which partial hypotheses will turn out to be the best. But developing all good partial hypotheses can be computationally expensive, especially if the cutoffs are set conservatively, allowing more hypotheses to continue. The Reggie system is able to afford this approach, in part, because it uses general world knowledge to bring additional constraints to the search. In the next two chapters, the modules that invoke world knowledge, grouping and occlusion interpretation, are presented.

Chapter 2

Grouping

2.1 An Introduction to Grouping

Grouping is the process of using general knowledge about objects in the world and about the imaging process to find collections of image features that are likely to be relevant to a specific vision task. There are many ways to perform grouping, and a wide range of vision tasks that may benefit by it. For the task of recognizing modeled objects, grouping is particularly desirable. There are at least two reasons for this. First, as discussed in the previous chapter, the combinatorics of forming all possible feature subsets and matches are explosive. The recognition task therefore has a great need to form relevant subsets more efficiently. Second, the additional knowledge brought to the process can improve accuracy[Jacobs 89], as discussed in Chapter 1. Both of these contributions are contained in a simple constraint that would be ignored by blind matching. It is this: a set of image features must all come from a single world object in order to correctly match parts of a single modeled object. General knowledge about the world is quite appropriate for finding likely single-object groups of image features. Because the grouping analysis is performed without reference to any specific model, it does not face the explosive combinatorics involved in matching. The grouping module's reduction in the number of possible feature combinations greatly relieves the burden on the later matching task. In short, Grouping is a way to use a great deal of match-independent information before resorting to match-dependent constraints.

Grouping is not Segmentation

If the task of grouping were to find all the features that come from each separate object, however, it would be very difficult indeed. One of the early goals of vision research was to segment the image completely into separate whole objects. Despite the

apparent simplicity of this task, it has proven to be virtually impossible to automate in an image-driven manner. Further, the concept of "separate objects" is not well defined. Is a flowerpot part of the plant object? The answer depends on the vision task, and even then may be ambiguous or allow overlapping interpretations.

Fortunately, there is a much less daunting form of image interpretation that proves to be quite useful for model-based recognition. If small groups of features can be found such that at least one group comes entirely from the object of interest, then grouping can be combined with verification to create a successful recognition system.

- Groups do not have to include the entire object, but only enough of its features to find the pose. (If they naturally contain more than the minimum, the pose can be found with greater accuracy.)

- It is not necessary to find all of the many possible groups on an object, because each group will be verified using the rest of the model. At the bare minimum, only one group need be found.

- The groups may overlap, thereby representing several interpretations of the image, because each hypothesis is considered independently. This is useful when faced with ambiguous grouping.

- Many extraneous groups can be tolerated before grouping becomes too inefficient, in comparison with all possible combinations of small sets of elementary features.

The above allowances are provided by the recognition modules, as indicated. There is one more important allowance that is derived from the interpretation module, described in the next chapter.

- A group may contain some features from occluding objects in addition to the main object, because occlusion will be explicitly interpreted.

Together, these effects make the grouping task much easier than complete object segmentation. They open the door to a wide variety of grouping methods. In practice, grouping has significant potential to reduce the number of considered groups and yet support reliable recognition, as demonstrated in this thesis and in [Lowe 87, Jacobs 89].

2.1.1 The Role of Grouping in Model-Based Recognition

The need for grouping arises from the difference between the amount of information provided in a single "elementary" image feature, and the amount required to form

a testable hypothetical match between model and image. The information deficit will depend on the choice of features and on the method of testing, but it persists. In order to be robust and repeatable, especially under various lighting conditions, features are extracted from intensity edges, and so are often based solely on the shape of curves in the image. Because the feature must be detectable under a wide range of viewpoints, only extrema of shape are typically used (straight lines, points of high curvature, points of inflection). Because the influence of occlusion must be minimized, features are local. These considerations conspire to restrict elementary image features to roughly two to four degrees of freedom. That is, a match between a single image feature and a single model feature provides only two to four independent equations to constrain the pose. This is not a formal result but an empirical bound that vision researchers have been unable to surpass.

Since the pose of a three-dimensional object has six degrees of freedom, a match of a single image feature to a model feature is not enough to solve for the model pose. In order to use alignment to find further matches, the pose is required. Therefore, small groups of elementary features are formed. Each group has enough information that, when matched with a model group, the pose is fully constrained. With point features, for example, groups of three points suffice.

It is tempting to say that the groups are simply larger features. Indeed, features must comply with the grouping rules. In addition to having the properties of pose-invariance and detection stability, features must summarize pixels of the image that are likely to come from a single object and must be extracted based on general knowledge of the world, not a particular model match. But groups differ from elementary features in two ways. First, groups may overlap— more than one group may contain a particular elementary feature, whereas elementary features themselves almost always summarize independent aspects of the image. Second, groups require internal matching. When an image group is matched to a model group, the internal correspondences of the features may be made in various ways. So groups have some different properties than elementary features, but can also be treated as image units with enough degrees of freedom to constrain the pose.

2.1.2 The Background of Grouping

Groups have traditionally been formed using various unmotivated heuristics, often as a part of feature detection. Specific approaches are discussed in section 2.5. Only recently has grouping been identified as an important operation in its own right. The theory of grouping provides a way of evaluating any particular grouping method, simply because it names the goal of grouping: to find features from a single object. Knowing this goal, the effectiveness and shortcomings of any grouping approach can be estimated, either empirically or through approximate geometric arguments. In

this way, grouping is elevated from simply a search-pruning technique to a first-class vision module, similar to feature extraction but separate from it.

Elementary Grouping Cues

Any relationship among features may be used for single-object grouping if it is more likely to occur when the features all come from the same object than if they come from different objects in the image. Listed below are some basic examples to illustrate this criterion. Many are mentioned in psychology literature (as early as [Wertheimer 23]), and some have been explored in computational vision (as discussed below). As simple as they sound, some are fairly powerful: features with certain properties can be much more likely to come from a single object than are random collections of features. However, not all have been tried, either in psychological tests or in vision systems.

Proximity: Features are close to each other.

Edge Connectivity: Features are derived from the same sequence of connected edge pixels.

Cotermination: The features end near each other.

Colinearity: The features lie along the same line or smooth curve.

Parallelism: The features are parallel.

Symmetry: Two feature sets have similar shape, either reflected about an axis or under various combinations of rotation, translation and scale.

Convexity: Features form a convex polygon when connected by imaginary lines.

Grey-level similarity: The features are extracted from or near regions of similar intensity, or intensity patterns, or intensity distributions.

This section has introduced the concept of grouping in general. Related work is reviewed in section 2.5. In the next section, I proceed to the specific feature relationship used by the Reggie system.

2.2 Region-Based Grouping

Region-based grouping is a new method of forming groups that has been developed as part of this thesis. In this section I define region groups, discuss what assumptions they are based on, and describe how they are formed.

Region-based groups capture one of the most basic visual clues we have about our world: that objects are solid, and occupy contiguous portions of the image, at least in a piecewise fashion. Because of this, features that surround an open region of the image are likely to come from a single object. An "open region" is a portion of the image in which there are no intensity edges, while the features that "surround" the region are edge-based features (currently line segments, but others could be used). In office scenes, open regions are almost always caused by a smooth unpatterned portion of the surface of an object, counting background surfaces such as walls and tabletops as objects. The open region of the image almost always comes entirely from a single object, because a transition from one object to another almost always causes a change in intensity sufficient to produce an edge that splits the region. Let the object whose surface forms the region be called the *region object.* The edges that bound the region may come from objects in front of the region object (occluding objects), or may be the outer edge of the region object, or may come from changes in surface orientation or coloring within the region object. In the later two cases, the edges belong to the region object and provide evidence about its 3D structure. They can therefore be used in later modules to apply geometric constraints in determining the identity of the object. In the first case, the edges belong to other objects, and should be ignored when making matches to model edges. The job of determining whether the surrounding edges belong to the region object or to occluding objects comes later, and is the subject of Chapter 3.

From this straightforward analysis of region formation, we can see that region grouping is based on some assumptions. As stated, it depends on objects being uniformly colored and having smooth surfaces. If objects with significant visual texture are to be recognized, a module to find contiguous regions of similar texture would be required, but the concept of region grouping would still be applicable. The features grouped by regions will still be likely to come from a single object. If groups are also formed in the model, however, then the stability and repeatability of the groups becomes an issue. We might expect region groups to be affected by highlights or shadows that cause edges other than the expected ones. Also, due to chance alignments or lighting, expected edges may not be detected in the image. Edge and feature instability can be a serious problem at times. Instabilities affect all recognition strategies, but region grouping is particularly susceptible since it tries to reduce the number of groups so drastically. Fortunately, there are two counteracting effects already mentioned: that only one group from the object need be successfully found, and that groups are explicitly interpreted for some kinds of feature instability, as described in the next chapter.

Another implicit assumption is that regions, and the edges that bound them, are well defined. Unfortunately, they are often not well defined, and this is the primary challenge in finding regions. Rarely do edges form a complete connected closure

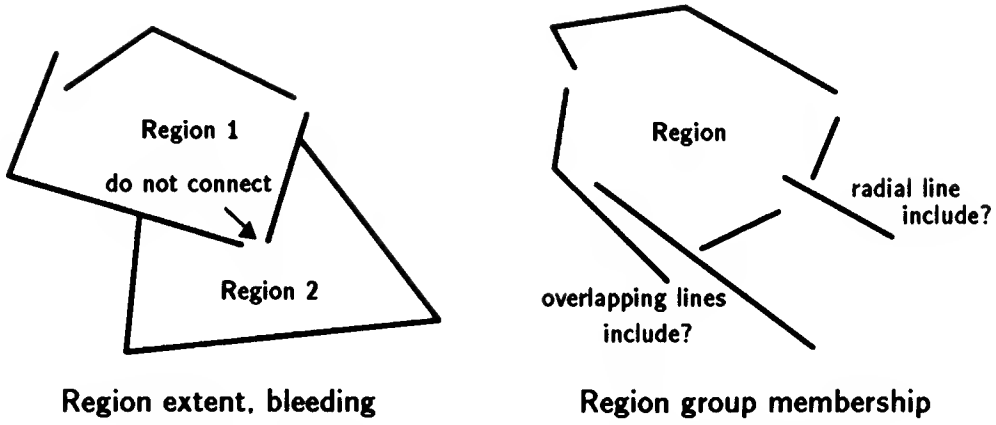

Figure 2.1: *Difficulties in determining region groups.*

around a region. There are almost always gaps in the edges that allow one visually distinct region to be connected, or "bleed", to others (see Figure 2.1). Some gaps should be closed, others should not, and in some cases both options should be pursued. The region grouping algorithm presented below quantifies the region bleeding problem using the ratio of the gap width to the size of each of the neighboring regions. In addition to the extent of a region, its bounding edge segments can be ambiguous. Edge segments that are radial to a region, for example, may or may not be included in the group. The algorithm described below tends to include all possible bounding features and postpones their elimination until the interpretive matching stage.

The solutions have proven to be successful. Two criteria are available to judge the output directly: the groups should tend to come from a single object, and they should be visually compelling to a human observer. Some results are presented in Section 2.4. From them it is obvious that grouping is more successful than forming all possible combinations of features—the set of region-based groups is much smaller and has a much higher percentage of single-object groups. But the output should not be judged only in isolation. Later recognition modules may be sensitive to some grouping imperfections, or may be able to compensate for them, so the ultimate criterion for forming groups must come from the success of the overall system. From experiments with several images and objects, at least one region-based group has always been found that supports recognition. Therefore, region-based groups satisfy two important system-wide criteria: efficiency and accuracy. Before presenting these results, I shall describe the region-grouping algorithm.

2.3 The Algorithm

The region-based grouping algorithm can be summarized in three steps. First, open regions are found using grassfire label propagation out from the edge features into the open regions. As the labels propagate out from the edge features, they fill in the open spaces, creating islands that are eventually swallowed up. Each island is taken as the center of a region. Second, the feature labels that propagated out towards each region center are collected. This is done by tracing back from the center along label boundaries. Limiting the extent of tracing is one of the main ways to control region bleeding. Finally, adjacent regions may be merged.

The result of this algorithm is groups of edge features that form what might be called "starfish-convex" regions (as distinct from convex or star-convex). The regions may be convex, but they may also have tapering limbs that may bend, as shown in Figure 2.2. Another way to visualize the family of region shapes is to think of mountains. Assume that all edges are rivers with elevation zero. Rising up from all rivers at a constant slope are mountain sides. The slope directly above each point in the river "belongs" to that part of the river. A region consists of all those edges that could be reached by traveling only downwards along ridges from a peak.

Pre-processing

The edge and feature detection used by Reggie is described in detail in Chapter 4 and is only summarized here. Before grouping, edges are found in the image (using Canny's algorithm[Canny 86]). The edge pixels are traced to form connected edge chains, or curves. Along each curve, straight line segments are found that approximate the curve to within one pixel, using a recursive split and adjacent merge algorithm[Pavlidis 72]. The line segments are taken as features. For 3D objects with straight sharp ridges, the 2D image edge segments are fairly predictable, and can often be matched one-to-one with model features. (Along smooth 2D curves in the image, the line endpoints are not predictable, but the lines compactly encode the location of the edges. This makes verification more efficient.)

Label Propagation

The line segments serve to group the edge pixels into pieces, each of which is very likely to come from a single object. (This is an example of grouping at the feature level.) Portions of curves with smoothly varying orientation, for example, could also be used for this purpose. Each piece of the edge is given two unique labels, one for each side of the edge. The labels are brushfire-propagated in the *label array*, which has elements corresponding to the image pixels. Initially, the labels are placed in

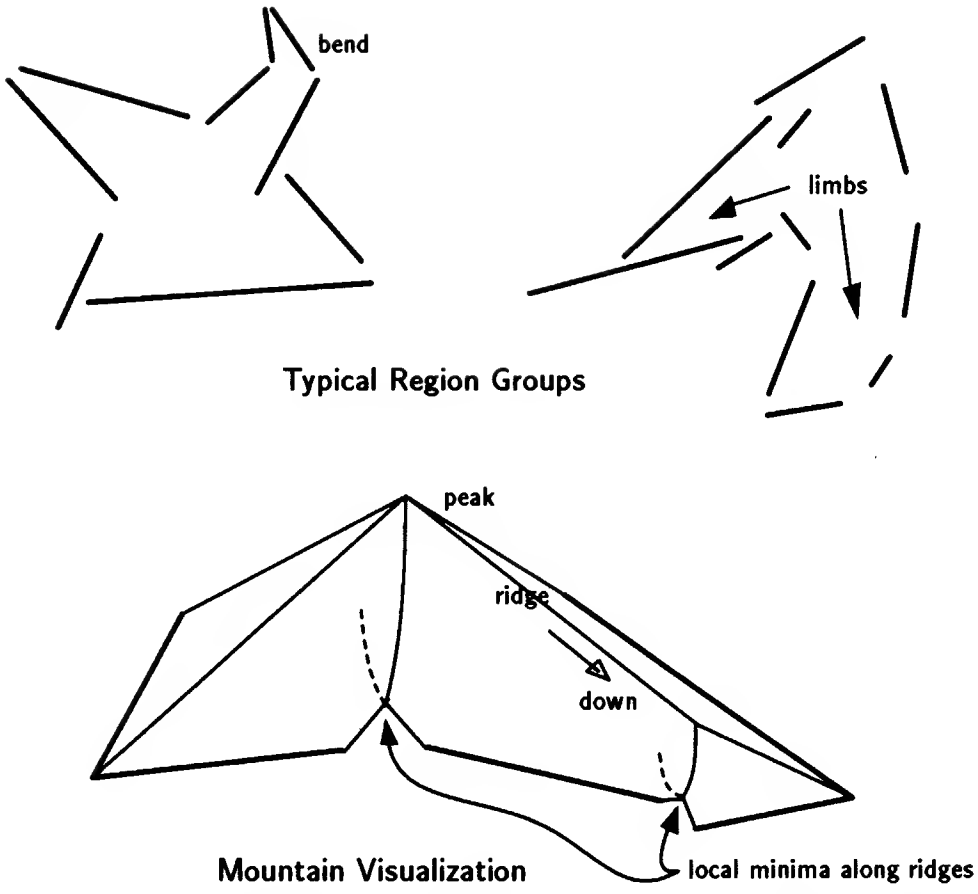

Figure 2.2: *Typical region groups, and the mountain visualization of the region definition. The trace continues downwards along ridges from the peak, branching, until it reaches saddle points.*

the pixels adjacent to the edges, as shown in Figure 2.3a. Then, on each iteration, every labeled pixel propagates its label to its immediate neighbors, if they do not already have a label (as shown in Figure 2.3b). The fact that labels only propagate to unlabeled pixels creates an important clogging effect, preventing labels from bleeding through gaps. Where multiple labels contend for a single unlabeled pixel, it is not important which one wins.

In addition to the label array, a *distance array* is kept. In it is recorded the number of iterations that were required to reach each pixel. This indicates, at every pixel, the distance to the nearest edge pixel as measured by the number of cells along the shortest path.

Initially, most of the image is open and unlabeled, and the open regions are largely connected to one another through gaps between edges. As the labels propagate, the image is filled in from the edges towards the centers of open regions, as shown in the sequence of images in Figure 2.4. As labels from different edge segments meet, connections between open regions are pinched off, forming islands of unlabeled regions. The islands continue to split or shrink until finally they are filled in. At the last iteration before they disappear, they define a local center of a region. The island centers are also local maxima in the distance array.

When the last island (and so the entire image) has been filled in with labels and distances, iteration terminates. The label array looks like Figure 2.5, where each color is a different label, but colors are reused. Note that color patches in this figure are not regions, they are the area of the image associated with a particular edge segment. The distance array is shown in Figure 2.6. The entire distance array is searched for local maxima. Since islands are not necessarily single pixels, the search is actually for connected blobs of pixels with the same distance value, all greater than the surrounding neighbors. Each of these local peaks is a potential region group.

Label Tracing

After the region centers are found, the group of edge features associated with each region center must be determined. Some of the labels of the surrounding features will have reached the region center, but others will have been "squeezed out" along the way (see Figure 2.7). Since these labels represent valid surrounding features, they must be recovered. To do so, a branching trace is performed as the second major phase of region grouping. Starting at each center, the trace proceeds along boundaries between differing labels. When it comes to an intersection of three or more boundaries, the trace splits off recursively, following each branch in turn. All unique labels are collected along the way. Care is taken to preserve the circular order of the labels around the center as they are recovered. (The circular order will prove to be very helpful during matching, as discussed in Section 3.1.)

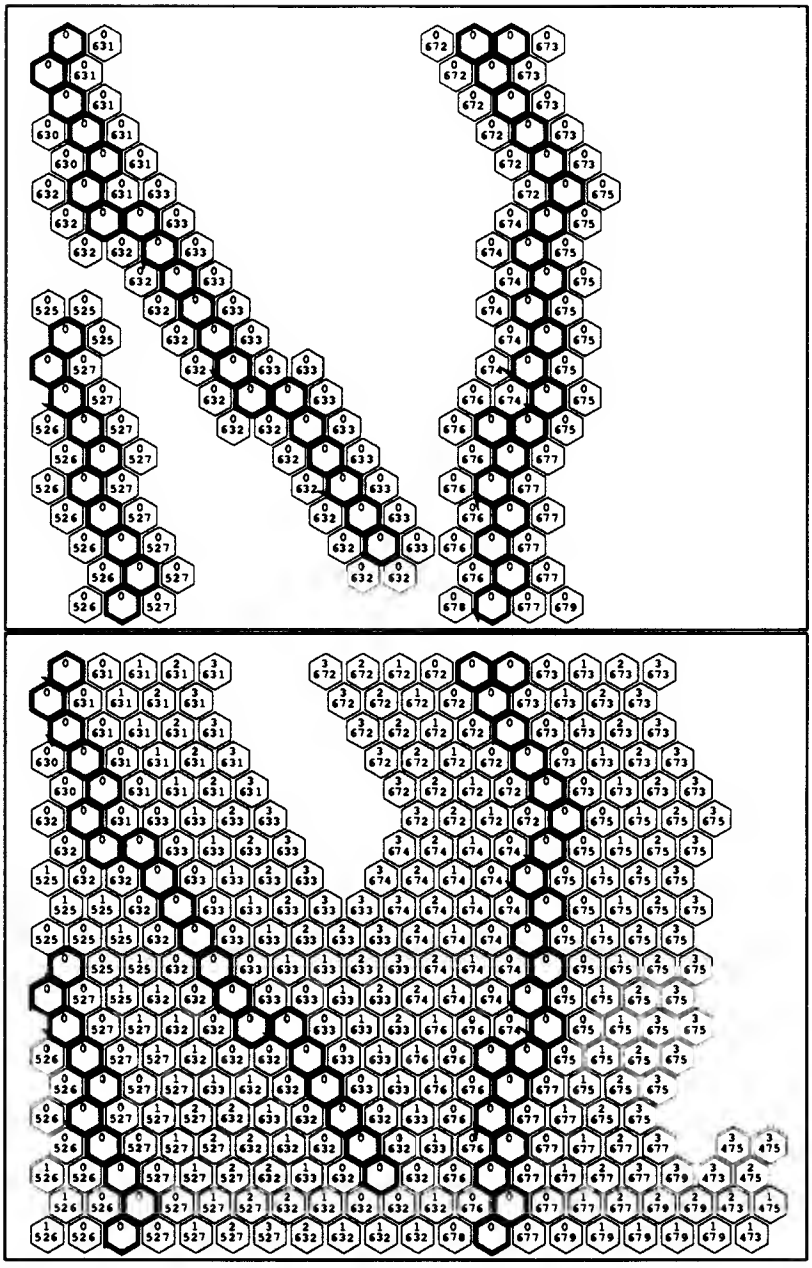

Figure 2.3: *a) Labels are assigned on both sides of edge segments. In the center of each cell is its label, at the top is the distance to the nearest edge. Note that different labels are given to each straight segment of an edge, as determined by the line-finding module. (The line segment endpoints are not indicated in this figure.) b) Labels propagate outward like a brushfire, passing their label to the nearest neighbors on each iteration. This is the state after three iterations.*

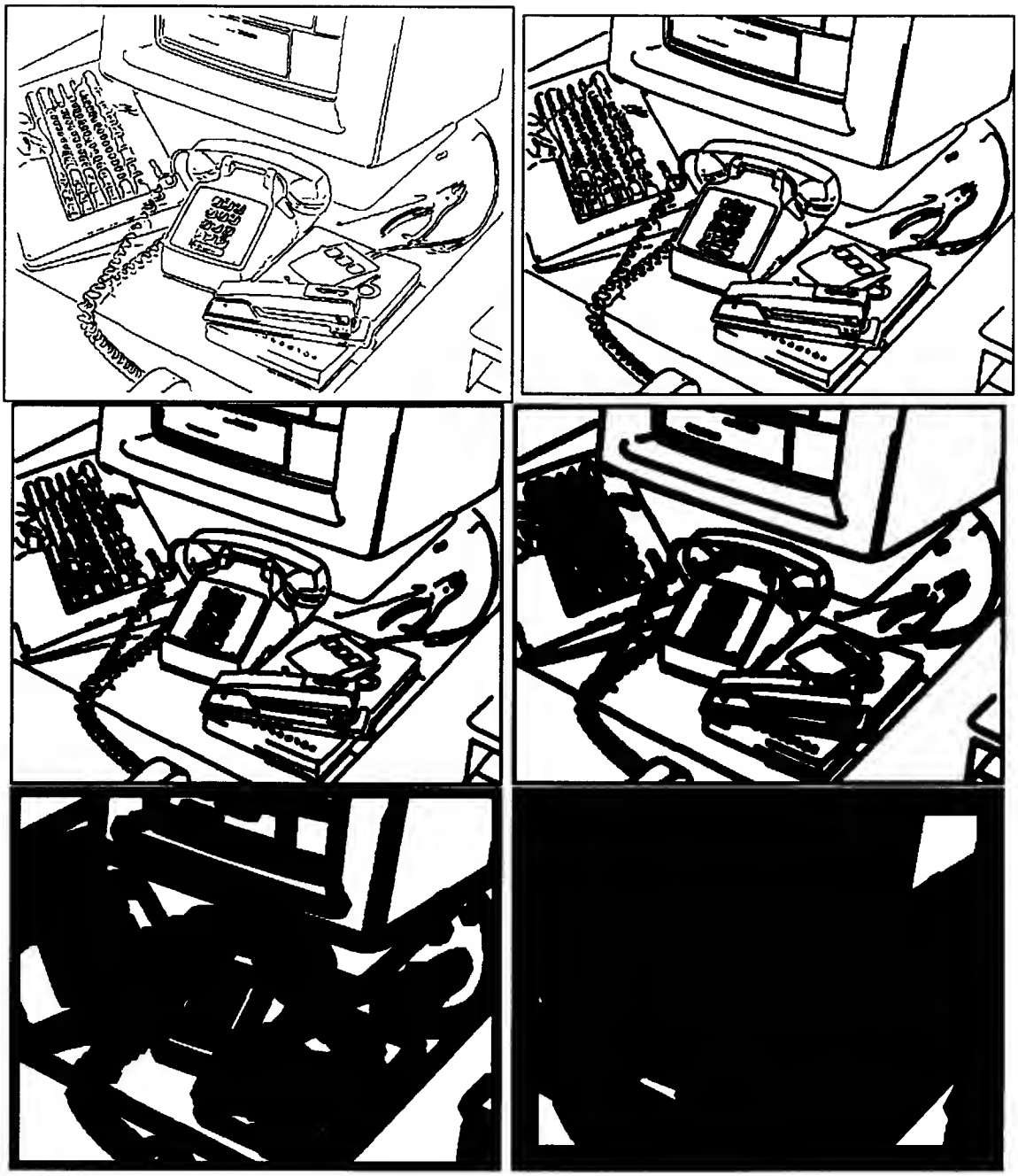

Figure 2.4: *Label propagation as seen in the whole image. Regions are initially connected in many cases through small gaps. As labels propagate out from edges they create islands of unlabeled regions. The islands continue to shrink, and define region centers just before they disappear. Shown are the edges, the initial label assignment, and after iterations 1, 3, 10, and 30.*

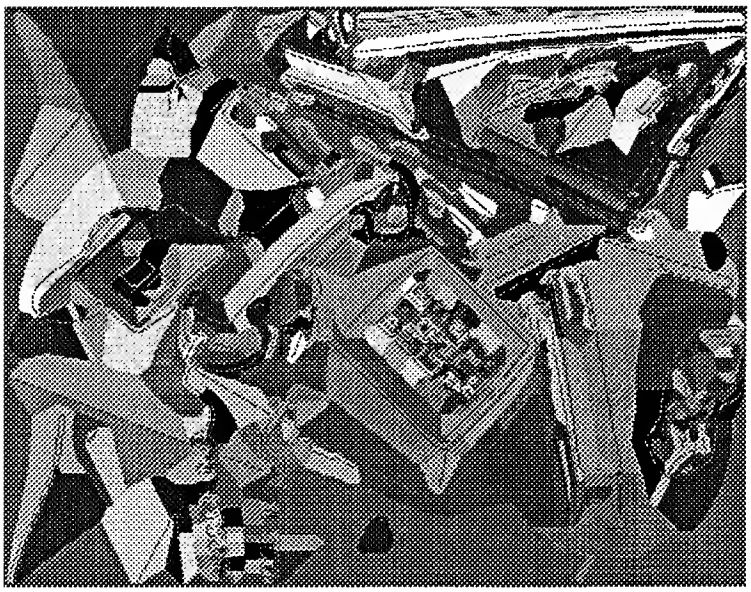

Figure 2.5: *The edge segment labels after propagating. For display, labels have been assigned random colors.*

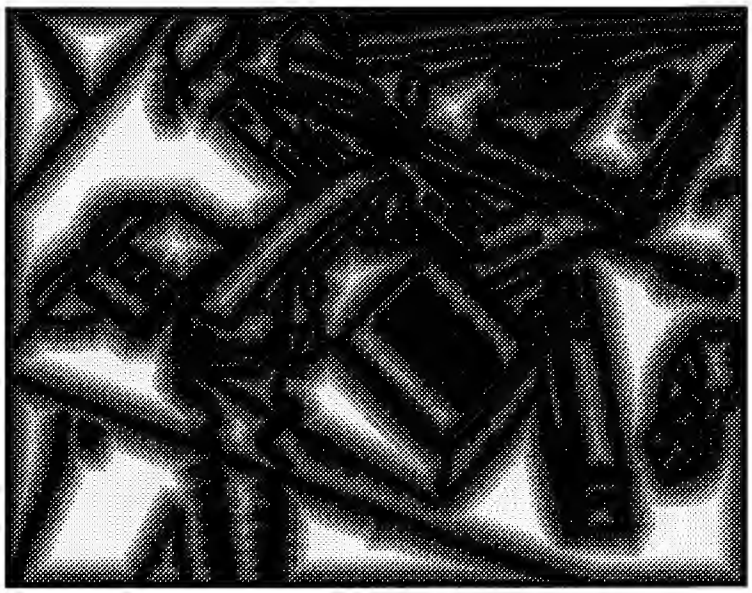

Figure 2.6: *The distance array. Lighter areas are further from edges.*

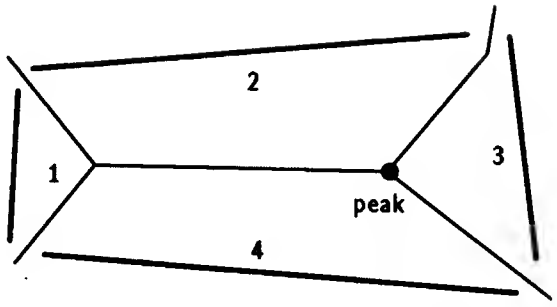

Figure 2.7: *The label 1, though it contributes to the peak, has been squeezed out by 2 and 4 on the way to the peak. The label boundaries must be traced to recover these labels.*

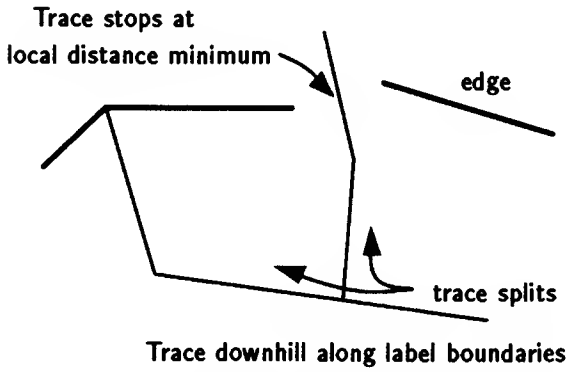

Figure 2.8: *As the branching trace proceeds along label boundaries, it reaches local minima in the distance array at gaps between edges.*

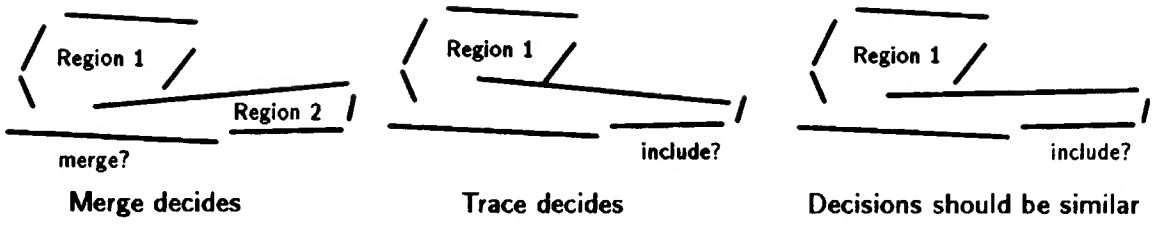

Figure 2.9: *Region formation in corridors. Because corridors are often visually distinct from the regions at their ends, the trace will travel only a limited distance along it. This cutoff should be consistent with the region merging criterion, however, because the same situation with only slight alterations will be a merging decision.*

If this trace continued unchecked, it would quickly enter into the territory of other regions. Instead, the distance array is used to limit the trace. As the label boundaries are followed, the distance array is referenced at those pixels. Tracing may only proceed *downhill* from the peaks in the distance array. That is, it may only proceed (along label boundaries) from the centers *towards* edges, not away. When a trace comes to a gap between two edges, as in Figure 2.8, it encounters a saddle point in the distance array. The label boundary between the different edges continues in the direction across the saddle such that the path is a local minimum, and hence the trace stops. (In addition, the trace will be stopped if it loops back on itself, as it might on level label boundaries.) The effect of downhill tracing is to group only edges that are getting closer as one travels away from the center. It is one of the most significant properties of the region-based groups.

But even this criterion does not completely solve the region-bleeding problem. As shown in Figure 2.9, some region shapes do not seem like single regions to most human observers, yet the trace would not be terminated in this case because the distance stays roughly level along the trace and does not reach a minimum. Hence, all the edge features would be grouped together, counter to visual intuition. If the distance is always decreasing, the region is tapering down and is visually acceptable as a single region. Or, if the distance reaches a minimum, the region will be pinched off appropriately. The problem comes only from the borderline case. The solution is to limit the extent of the trace along a corridor. As the trace proceeds, the number of steps taken at a constant level is recorded. If it exceeds the width of the corridor, the trace is terminated.

Region Merging

The limits on label growth and on tracing serve to prevent regions from leaking into one another. But, for various reasons, there may also be too many local maxima in

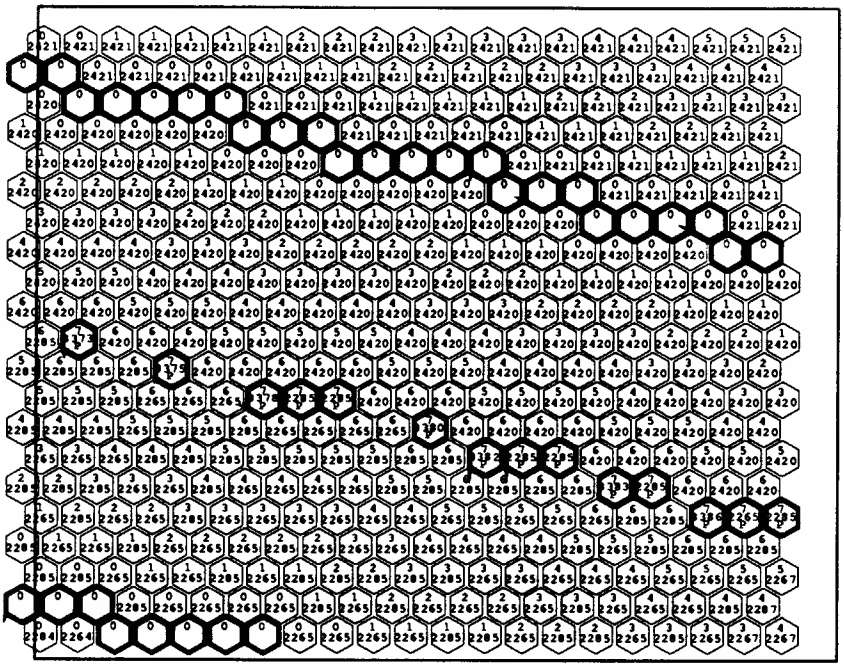

Figure 2.10: *Due to the discretization of the image, small anomalous peaks may be formed along ridges.*

a single region, and it is important to merge regions sufficiently.

There are at least a few clear cases where regions should be merged. First, two separate peaks may trace back to exactly the same surrounding features. In that case, they represent the same group, and so are not duplicated. Second, as different labels grow and meet, creating a label boundary, tiny islands (which become local peaks) may be formed due to the discretization of the image. An example is shown in Figure 2.10. If the process had been performed analytically with a continuous plane, these peaks would not appear. They do not reflect any important geometry of the edges, and so are merged into the peak uphill from them.

In addition, there are some cases that require subjective judgement. In Figure 2.11b, one might say that there is one region or two. In borderline cases such as this, either outcome is acceptable. But a threshold must be set somewhere, since Figure 2.11c is probably two regions, while Figure 2.11a is probably one. Fortunately, the distance array provides a means for quantifying the decision, if not objectifying it. The distance value at the trough between the regions indicates how tight the squeeze is between them. A threshold on the ratio of the trough height to the larger peak height is used to decide whether to merge the regions. The threshold is set subjectively.

In another subjective case, any group that is a proper subset of another group

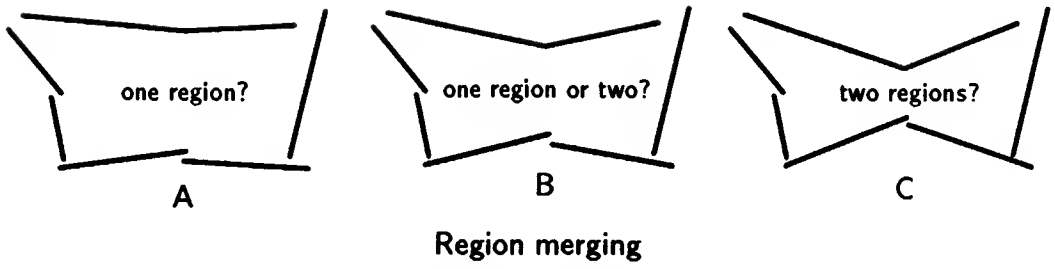

Figure 2.11: *Some threshold is required, wherever it is set.*

might seem superfluous. However, there may be value in keeping the smaller group in addition to the larger, since the larger group may contain incorrect extra edge segments that would disrupt proper matching. Despite this, all subsets are removed in the current implementation.

2.3.1 The Resulting Region Groups

The region groups resulting from this algorithm have some notable properties. First, though the boundaries of the regions are never explicitly defined, it is clear that the regions do not overlap. They do not cross over edge segments, and traces that terminate at an edge gap create a virtual edge boundary. The only possibility for overlapping regions comes in the merging stage, where two regions may be preserved distinctly in addition to creating a region from their union. Currently, the algorithm does not preserve two distinct regions if they are merged (though it might be desirable to do so in some cases). Nonetheless, there is a large amount of overlap among the sets of features in the region groups. Most edge segment features belong to a different region group on each side. And, though regions are distinct, they often share a feature near their boundary.

As shown earlier in Figure 2.2, regions are not necessarily convex. Each branch of the trace down from a distance peak will lie between edges that are growing closer as the trace moves down, but the trace may curve. Even if each trace branch were convex, convexity is not enforced between different branches.

It may be elucidating to note that the traces form a discrete approximation of the nearest-neighbor Voronoi diagram for the edge segments. In the Voronoi diagram, each "source" (usually points, but in this case edge segments) is allocated a surrounding cell, such that every point in the cell is closer to that source than to any other source. Since the traced label boundaries lie along locations that are (approximately) equidistant from the edges, they form similar cells. A region does not correspond to a cell— they are collections of pieces of Voronoi cells. The cells are broken at points where the boundaries are closest to the source.

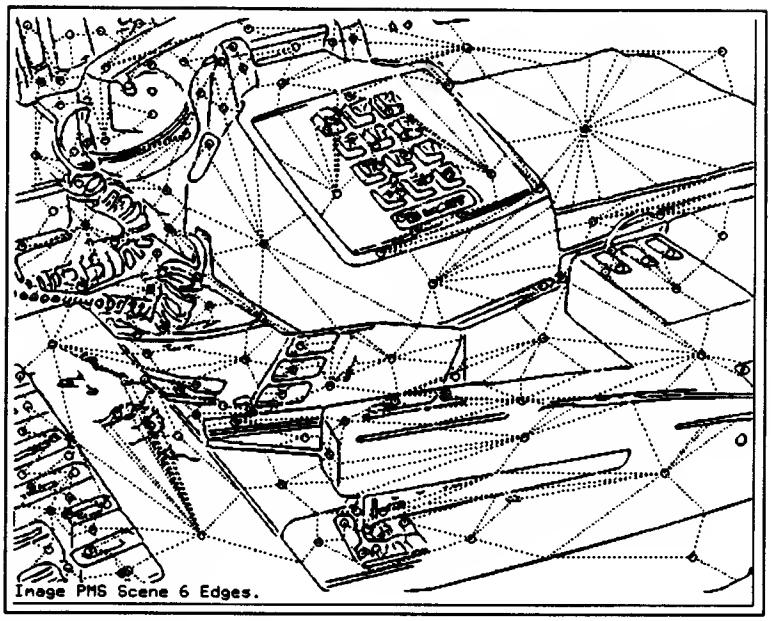

Figure 2.12: *These are the region groups found in a typical image. Each group is represented by a circle with dashed lines to the edge-segments that are its members. Each dashed line leads to the middle of a line segment, though the segment endpoints may not be visible.*

These observations provide some conceptual insight into the kinds of region groups that might be formed. A more compelling justification comes from the performance of the algorithm in the recognition system.

2.4 An Evaluation of Region Grouping

This algorithm produced the region groups shown in Figure 2.12. In the figure, each group is indicated by a circle at the center, with dashed lines to the mid-points of the line-segment features in the group. The region areas are not directly depicted, in order to emphasize the fact that the primary goal of region grouping is to group features, not to form regions.

Are these groups acceptable? As mentioned above, any grouping system should be both efficient and accurate. That is, it should form a small number of groups in a reasonable amount of time and still include enough single-object groups to enable recognition.

The efficiency of the entire system will depend additively on the speed of the grouping module itself. However, it will depend multiplicatively on the number of

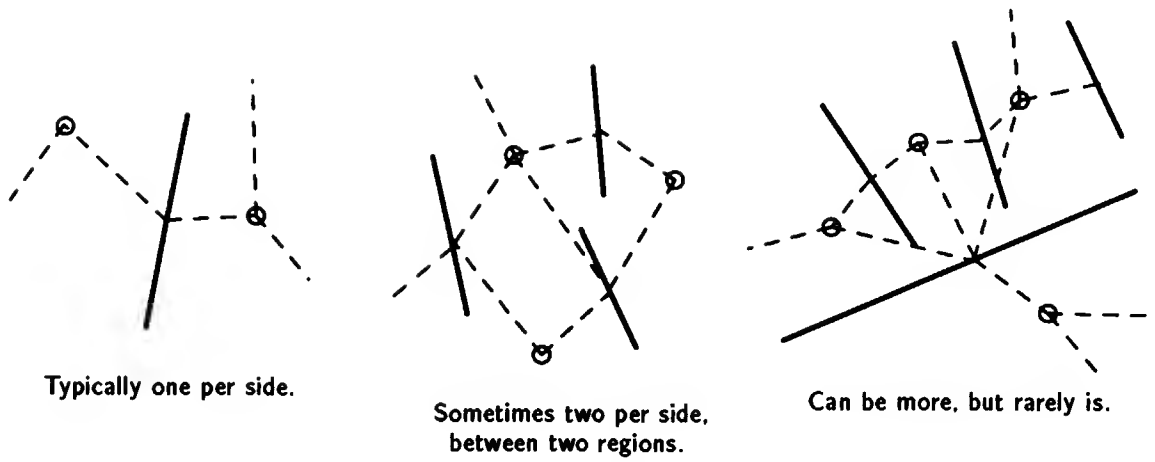

Figure 2.13: *Since the number of groups to which a line belongs is generally bounded and not dependent on the total number of lines, the number of groups is generally a linear function of the number of lines.*

groups produced from the image, $P(N)$, since all $P(N)Q(M)$ combinations of groups are tried (where $Q(M)$ is the number of groups in the model). I now argue that the number of region groups is roughly linear in N, the number of image line features. The number of groups is equal to N multiplied by the average number of groups in which each line feature is a member, n_g, divided by the number of lines in a group, n_l. At worst, n_l is one, so it remains to be shown that n_g is constant, independent of N. From the images, and as shown in Figure 2.13, we can see that there tends to be one region group on each side of a line feature, plus occasional overlap between adjacent region groups. If each line only belonged to two groups, then there would be a linear upper bound on the number of groups. Allowing for the possibility that neighboring regions share the features between them, a line can belong to at most four groups—still showing linear growth. The argument is not tight, however, because a situation can be manufactured in which any one line can belong to as many groups as desired. In practice, the frequency of this situation is rare, and no more than linear in the number of lines.

Empirically, for twelve images, the number of lines (of length eight pixels or greater) is plotted against the number of region groups (of size four or greater) in Figure 2.14. Rather than linear growth, it indicates that the growth is not particularly sensitive to the number of lines, and that both numbers remain roughly constant for typical images. More importantly, the constant of proportionality is very promising. Instead of tens of millions of possible triplets of image features, region grouping produces only about seventy groups. Also, each group contains more than just three features (7.5 on average), providing more information for pose-solving. (Larger groups

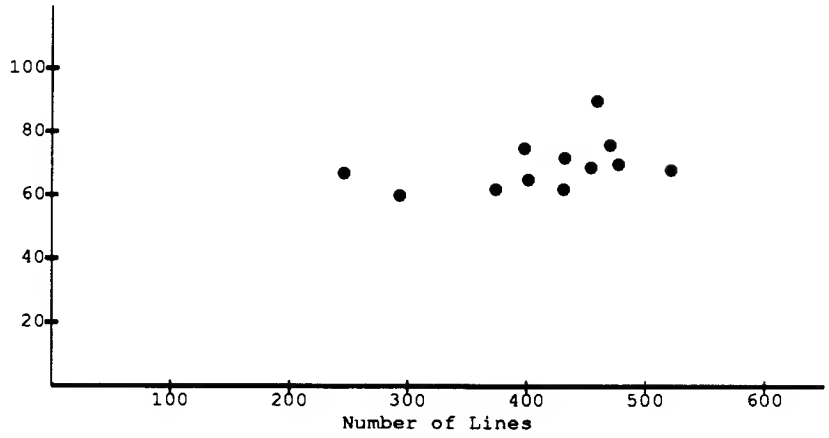

Figure 2.14: *The number of groups (with four or more members) found in an image, plotted against the number of lines (of length greater than eight pixels) in the image.*

also cause more possible interpretations during matching, which increases flexibility but may also increase the time spent per group.)

In order to evaluate the reliability, line features were hand sorted in one image, by object. (Image 3 was randomly selected, out of the images shown in Chapter 6.) The distinct objects included were the mouse (without the tail), the stapler, the phone, each book, the wad of paper, and the table, which together accounted for 85 percent of the image lines. Three quantities were totalled: the number of groups whose members were entirely from a single object, the number of objects with at least one group completely from it, and the average number of single-object groups per object. The results were remarkably good. Out of ten objects identified in the image, only one (the table—a background object) was without a group entirely from it. The average number of single-object groups per object was 8.3, when only one is needed. Out of all 150 groups formed, 83 were single-object groups.

Total number of groups	Number of single-object groups	Total number of objects	Objects with at least one group	Average number of single-object groups per object
150	83	10	9	8.3

If the recognition algorithm considered all matches between the model features and the image features within each image group, these results indicate that region grouping would provide extremely high accuracy, and almost perfect efficiency compared to forming all combinations of features in the image. However, because there

are too many ways to match each image group to the entire model, the model features are also grouped. The model groups are determined during model formation from the region groups found in images of the model object in isolation. This process is semi-automated, and is described in Chapter 5.

In order to provide sufficient accuracy in this matching scheme, the image groups must have sufficiently large intersection with the model groups. To measure this, the correct correspondence between image and model features was hand selected for Image 3, and the Mouse model. In Image 3, the mouse is almost entirely visible, but is small enough that many model features were omitted. It is fairly typical of the images. Image 3 has six groups entirely from the mouse, and the mouse model has fifteen groups. For each of the image groups, the largest intersection with any model group was found (via the known correct matches). Those intersections were of size 1/3, 2/2, 3/3, 3/3, 4/5, and 8/9, where the denominator is the total number of lines in each image group. Since three lines should be enough to solve for the model pose, four of the six image groups were sufficiently similar to the model groups to support recognition. Only one is needed.

This indicates that the region groups were sufficiently stable to maintain accuracy while drastically reducing the matching combinatorics in both the image and model. In the next chapter, the combinatorics of matching features within groups is challenged.

2.5 Related Work in Grouping

Many vision systems have implicitly used grouping to reduce the number of match hypotheses, as discussed in Section 2.5.2, below. But the explicit theory of grouping is most directly related to this thesis, as described in the next section. It draws on psychological theories of perception, and only recently has been explicitly applied to computational vision. In Section 2.5.3, some related work in region formation is discussed.

2.5.1 Grouping Theory in Psychology and Machine Vision

In psychology, concepts related to grouping have been part of the understanding of human perception for many years. Since the 1920s, Gestaltists have argued that pieces must be brought together to be understood. Though they detailed several criteria for grouping, as observed in humans, they did not offer any theory of how grouping was to be implemented.

Perceptual organization has been observed and analyzed in humans for many years, from [Wertheimer 23] to a collection of papers entitled *Perceptual Organiza-*

tion [Kubovoy and Pomerantz 81]. For the most part, psychology researchers have attempted to capture a fundamental principle of grouping that is consistent with the performance of humans. For example, [Hochberg 57] defined a principle of simplicity, based on minimizing the amount of information required to describe a scene.

In [Marr 77], Marr proposed a theory of machine vision that included some concepts related to grouping. He claimed that an interpretation of the image should be derived, called the "full primal sketch", containing groupings of curves and other features. Lines would be grouped on the basis of curvilinearity, colinearity, and parallelism, and regions of similar tokens would be grouped on the basis of properties such as size. Despite the importance of the link to computational vision, no successful implementations were developed based on this kind of grouping.

[Witkin and Tenenbaum 83] proposed a non-accidentalness criterion for grouping. They report that grouping is accomplished in humans without any reference to high-level knowledge of the content of a scene, and point out that grouping is important to many computational vision problems, for many of the same reasons that segmentation would be desirable if it could be achieved. However, they did not develop a recognition system.

It was not until 1984 that grouping was explicitly applied to a model-based recognition system, by David Lowe [Lowe 85]. Lowe demonstrated that the concept of perceptual organization could be gainfully melded with search-based recognition. He outlined his own non-accidentalness criterion for grouping, and described how finding single-object groups and inferring some 3D relations could aid recognition. In his implementation, he grouped parallel and coterminating lines to form minimal hypotheses, matched groups to model features, solved for the pose, and verified the hypotheses by projecting the model into the image.

In [Jacobs 88], Jacobs presents a computational theory of grouping, based on the probability that two sets of edges are produced by the same object. He then presents a geometrical and empirical analysis of why certain grouping criteria work, derived directly from the image formation process and a simplified world of objects. From the analysis, he proposes a grouping strategy based on convexity. His approach is to find primitive convex groups, and then combine these into larger groups based on their distance and relative orientation.

[Huttenlocher 88] forms groups of lines by their edge connectivity. The groups can be found in linear time by moving along the lines from an edge curve, and keeping every sequence of length three. From the intersections of the three lines, three points can be found to solve for the pose. Longer lines and lines from the center of the image are used first, in a best-first search. The groups form the initial hypotheses for verification by alignment of the model with the image.

[Mahoney 87] has developed a parallel architecture consisting of simple locally connected processors, explicitly for the purpose of "image chunking". This concept

is closely related to grouping, and the architecture should provide an efficient means for implementing various perceptual organization operations. In particular, Mahoney plans to implement visual routines such as those proposed by [Ullman 83].

2.5.2 Grouping Techniques Implemented Incidentally

Various types of operations that could be called grouping have been a part of machine vision since the earliest systems. But most have been developed without an explicit investigation of the operation. The grouping operations were performed without justification, and often without acknowledgment that the goal was to bring together pieces that were likely to be from a single object. Still, the various approaches are of interest. The descriptions of related work presented below draw heavily on the useful and extensive reviews that can be found in [Huttenlocher 88] and in [Jacobs 88].

Even in the pioneering work of [Roberts 65], vertex features are grouped along the straight edge segments that connect them. This is still one of the most popular grouping strategies. Unfortunately, Roberts relies on finding complete convex polygons in the image in order to identify vertices, so the match and verification strategies are quite sensitive to occlusion and noise. Also, Roberts was only able to use convex polygons because the modeled objects were restricted to polyhedrons. The recognition method does not directly generalize to other kinds of models. However, this is an example of how general knowledge about the library of models can be used to develop a grouping strategy, even though specific matches to models are not used during grouping. It is the general knowledge that the models come from a particular class of objects, polyhedra, that allows the vertex-edge-vertex grouping to be relied upon.

Roberts system also demonstrates how grouping can appear as part of search-pruning. Rather than explicitly forming the groups, an initial seed match between one image vertex and one model vertex is formed, and further matches are formed by following edges in the model and image. In this way, the grouping criterion, which depends only on the concept that connected vertices are likely to be part of the same object, is mingled with the correspondence search, in which image and model features are matched. It provides a criterion by which the search excludes many possible matches— image features not connected by edges to the seed feature will never be considered as matches to model features during hypothesis formation. (Those matches may be considered later, during verification, when the search is constrained by knowing the pose of the model in the image.)

Also mixing grouping and search, Bolles and Cain used a different grouping cue, proximity, in their Local-Feature-Focus system. Before recognition, they combine information about each type of feature in the model to create a search tree of the possible nearby features. They are able to use distances as effective constraints because of

the two-dimensional domain, in which distance is preserved under the transformation from model to image.

Several researchers have used generalized cylinders to represent objects, and grouped by finding ribbons or trapezoids ([Brooks 81] in ACRONYM, [Connell 85]). In these cases, the groups were considered features, even though the groups were constructed from lines found in the edges, not directly from the edges.

2.5.3 Related Work in Region-Finding

The general process of segmentation has been explored by many researchers, as reviewed by [Haralick and Shapiro 84]. Segmentation differs from region grouping in several important ways. First, region grouping is not directly concerned with determining the parts of an object, but only with finding likely single-object groups. Second, groups and their implicit regions may overlap. Third, they do not have to include the entire object, or any special decomposition of it. Fourth, grouping does not try to identify any properties of the image on a per-pixel basis.

Some shape representations are loosely related to grouping. The Voronoi diagram has been used as early as [Blum 73] in his Symmetrical Axis Transform (SAT). A related representation, Smooth Local Symmetry (SLS), was developed by [Asada and Brady 86]. Both were trying to capture the primary parts of objects, to produce a segmentation both of the image and of the object as well.

Chapter 3

Interpretive Matching

Once region groups have been found, they must be matched to model groups. There are many possible correspondences between image features and model features, even though groups have been formed. The search for correct correspondences can be divided into two parts: matching image groups to model groups, and matching the features within each image group to the features in each model group. In matching groups to groups, all combinations are tried by the Reggie system. However, in matching features to features, the interpretive matching module uses both knowledge about the world and about the model to efficiently prune the combinations tried. There are three main parts to this job, as diagrammed in Figure 1.3, and described in this chapter. These are briefly introduced here.

The first part of interpretive matching is *occlusion interpretation.* As described in the previous chapter, region-based grouping takes advantage of some general properties of objects to find features that are likely to come from the same object. But even groups that successfully contain several features from the same object often contain a few features from other objects. By far the most common reason for this is occlusion. Whenever an object is in front of another, we can expect that its occluding edge will be one of the boundaries of the region behind it. In that case, the image features derived from the occluding edge should not be matched to model features. The job of occlusion detection is to identify this situation by looking for clues in the image that indicate when one object is in front of another. In particular, if there is evidence that a feature is more connected to features outside of the group than those in the group, it may be from an occluding object. If it is ambiguous whether occlusion has occurred, both hypotheses may be explored.

Occlusion detection is implicitly an interpretation of relative depth of objects in the world, based on clues in the image. However, there is no attempt to assign depth values or determine any depth shape, only to hypothesize about whether parts of objects are in front of others. Further, there is never an attempt to form a globally

consistent interpretation of relative depth.

In addition to features from occluding objects, groups may contain features that come from the region object but are not part of the model, or are part of the model but are not stable. Detecting and accounting for *feature instability* is the second part of interpretive matching. The difficulty of detecting features repeatably is probably the second most important reason (after occlusion) that an otherwise correct feature match would fail. Lines may have roughly the right orientation, but the endpoints might move up and down the curve significantly. Lines viewed nearly end-on become short, and may not be detected. The orientation of short lines is harder to measure accurately. Smooth curves may be broken into line segments at arbitrary places. Or, a roughly straight line may be on the verge of being broken in the middle, and so will appear as two line segments in some images. If a one-to-one feature match is required, these feature instabilities can prevent the formation of a correct feature correspondence. But in some cases the instability condition can be detected, leading to alternate hypotheses about how to match the image features to the model features.

Grouping, occlusion interpretation, and feature instability detection are all accomplished without reference to the model information. They apply constraints derived only from our general knowledge of the world, or of the model library as a whole. Therefore, they do not suffer from the combinatoric explosion involved in matching image groups to model groups. But, of course, there are important further constraints that apply when specific model features are matched to specific image features. These are the most commonly used constraints in model-based vision. For the most part, they are geometrical; the shape of the image group must be consistent with some projection of the model group. Of these constraints, some can be applied without solving for the pose, and others are most easily applied after solving for the pose. Those that apply independent of the pose are called *invariant* constraints, and their application comprises the third part of interpretive matching. Pose-dependent constraints are left to the verification module, discussed in Chapter 4.

Invariant constraints include cotermination, colinearity, and parallelism. Where model features have these properties, the image features must also, or else the match cannot be correct. However, the application of these constraints must be interwoven with occlusion interpretation and feature instability correction. When occlusion is hypothesized, some of the group features may be partly or wholly invisible, and so the constraints may not apply. The job of implementing interpretive matching revealed that these kinds of interactions can be fairly complicated—the interpretation modules are not as independent as one would hope.

In the rest of this chapter, I describe the modules I have developed to accomplish the three parts of interpretive matching. The implementation is by no means exhaustive— I believe these concepts have significant potential for further work. However, as presented in Chapter 6, experiments with the entire recognition system show

that these modules are successful and clearly demonstrate the utility of interpretive matching.

3.1 The Match Search Framework

Interpretive matching begins by matching each image group to each model group, in all combinations. There is no overall attribute of a group to discriminate among the possible group matches. The number of features cannot be used, since occlusion or feature detection instabilities may cause features to be omitted in either a model group or an image group.

Given an image and a model region group, the task of interpretive matching is to find the feature correspondence. A primary constraint on the correspondence is that, within each region group, the features are in circular order around the boundary of the region. As discussed in the introduction, this is based on the assumption that objects are opaque.

The effect of this constraint is significant. If all combinations of feature matches between the groups were possible, the search tree could be structured as in Figure 3.1a. Each level represents the different ways to match one image feature to any of the possible model features. This has been called an "Interpretation Tree" by Grimson and Lozano-Perez [Grimson and Lozano-Pérez 87]. In their work, the sensory and model data are assumed to be of the same dimension (no projection), so dimensional geometric constraints (lengths and angles) can be applied to the matches as the tree is explored. The constraints are effective in pruning the search tree, eliminating incorrect match sets after only a few matches. However, in order to account for occlusion, a special branch is added to each level allowing the model feature to be skipped. Though the fan-out of the tree is not much increased by adding one branch, it is a branch that is not constrained. Consequently, it has the effect of opening up another whole level at each level. Accounting for occlusion greatly increased the number of nodes explored.

With the circular order of the features preserved, the tree can be redrawn as in Figure 3.1b. At the first level, the starting match is not known. A starting image feature is found that is not omitted under any interpretation. (When there is none, an ordered list of less attractive starting-features are tried). Each model feature is then tried as the first match. At each level thereafter, the search can only take one of three branches: form the match of the next image feature to the next model feature, skip the next image feature, or skip the next model feature. The tree therefore has a branching factor of three instead of $M + 1$.

The order of features by distance or connectivity has been used before to constrain the search[Ayache and Faugeras 86, Bolles and Cain 82]. However, in those domains

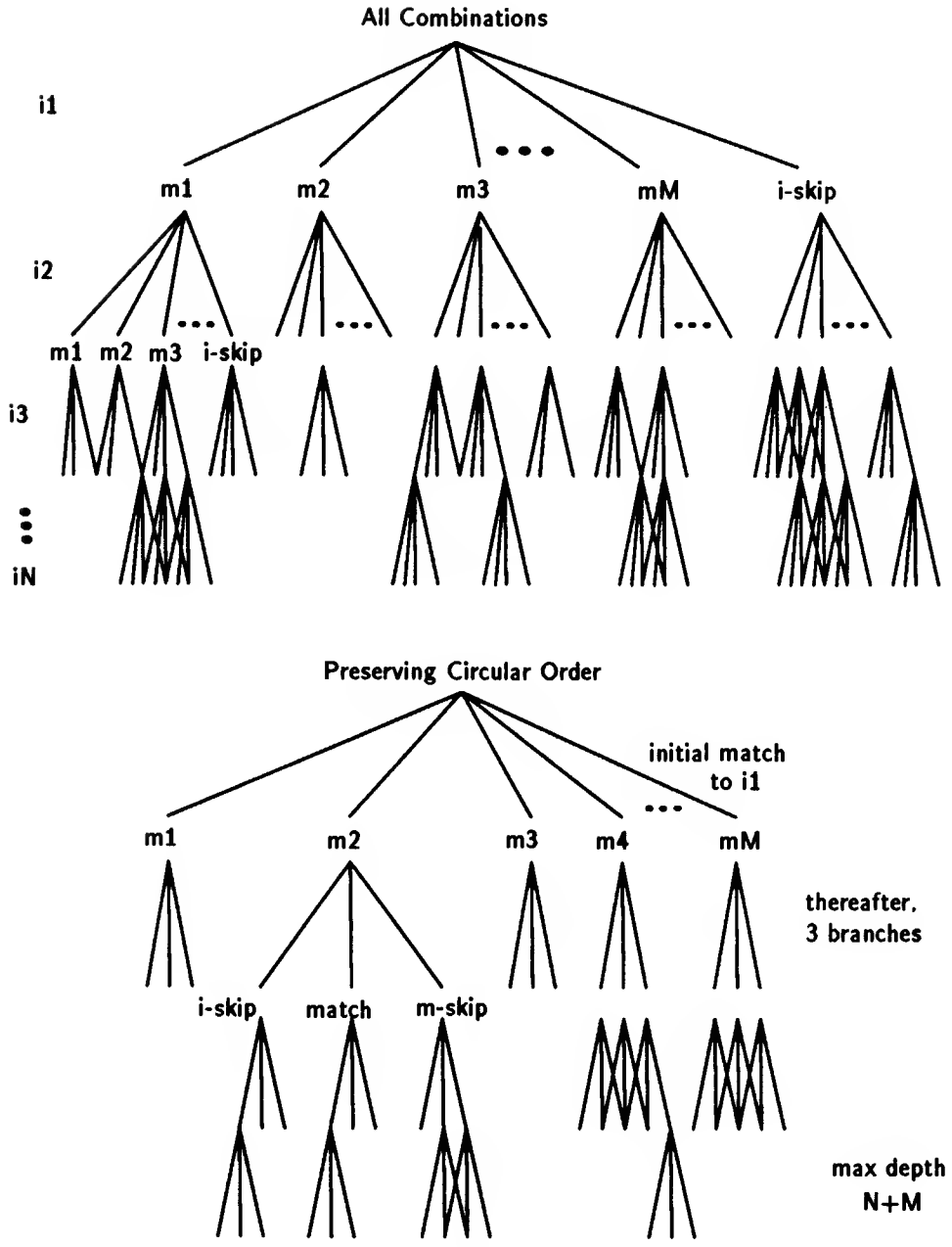

Figure 3.1: *Interpretation trees. a) All combinations of matches, with M-wise branches. b) All combinations with circular order preserved. Only the first level has M branches. At every branch thereafter has only three alternatives: skip the current image feature, match the current image and model features, or skip the current model feature.*

(2D images of 2D worlds, or 3D images of 3D worlds) there is no projection or scaling to alter distances in the model. Because of this, the pose is more readily determined and their use of distance and connectivity is part of verification, not part of initial match formation as it is here.

At each node, which branches are taken, if any, depends on the current interpretation of occlusion and feature instability. Because the interpretation state affects the direction of later branches, it is also possible for a single match-action (match, image skip, or model skip) to be split into several branches with different state, but this is very rare.

In addition to occlusion and instability, pose-invariant geometric constraints affect which branches are explored. Altering the pose can make almost any 3D model distance appear as any 2D distance in the image, and almost any 3D angle appear as any 2D angle. The only exceptions are the extremes: distances of zero and angles of zero and pi are preserved under projection. Hence, the pose-invariant constraints are limited to these extremes. When an interpretation violates a constraint, or runs out of image or model features, the branch is truncated at that point and none of its descendants are explored. A match hypothesis is formed from every match set that accounts for all model and image features without violating the constraints, and has enough matched features to solve for the model pose. Chapter 4 contains a description of the pose solution and verification process.

The matching scheme described above forms the framework in which the interpretation modules are invoked. The remaining sections describe the internal workings of the modules.

3.2 The Interpretation Modules

3.2.1 Elementary Feature Relations

All of the modules base their interpretation on elementary feature relations that are found when the line features are first created. They are the building blocks of interpretation, as they capture basic geometric properties of pairs of lines. The relations are associated with specific ends of the line features. They are detected using the following tests, as illustrated in Figure 3.2:

Cotermination Line endpoints are within 8 pixels of each other.

Colinearity The angle between line tangents is less than 10 degrees, and the distance between the endpoints is less than the sum of the line lengths. In addition, the gap between the line endpoints forms an imaginary gap line. The angle between the gap line and each of the lines is less than 10 degrees.

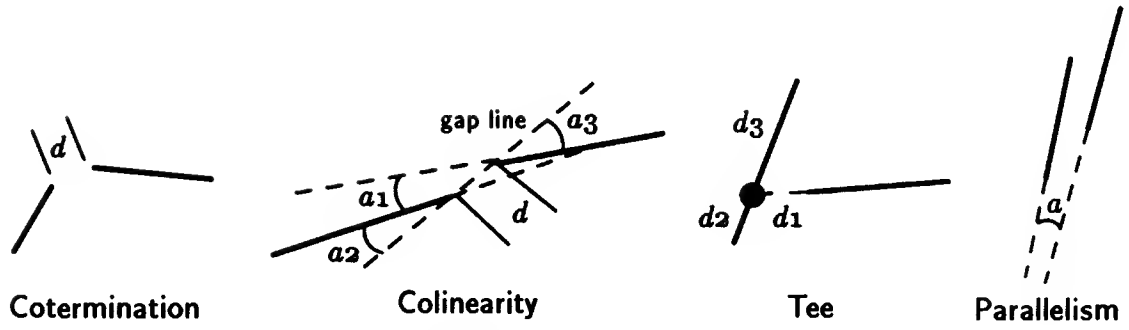

Figure 3.2: *Elementary line relations.*

Tee The distance from the endpoint of one line to its intersection with another is less than 12 pixels, and less than the distance from the intersection to either endpoint of the other line.

Each of the above relations includes a proximity bound. A fourth relation, parallelism, is found without a proximity requirement. To reduce the number of occurrences, it is only checked for all pairs of lines within each region group.

Parallelism Difference between line angles is less than 10 degrees.

The constants in these relations are set by experience, not by theory. However, the distance limits that are constant have to do with properties of the edge-detector, rather than perception. Perceptual distance thresholds are relative to the lengths of the lines, in order to account for relations occurring at various scales.

The relations are the basic building blocks for interpreting the image features. Before matching begins, for each group, an interpretation plan is constructed from the relations. Each plan has an entry for each feature. The entry summarizes the feature's relations to other features in the group, as well as including local occlusion and instability hypotheses. The image plans are used in conjunction with similar model-group plans to control interpretive matching.

Therefore, each interpretation module has two parts: before matching, the feature relations are interpreted to produce the relevant parts of the plan, and during matching, the image and model plans are compared to determine branching in the search tree. Each of the three modules (occlusion, instability, and constraint) are described in two parts, "Forming Plans from Feature Relations" and "Using Plans to Control Matching". Match control consists of determining which of the three branches to take at each node in the search tree. At each node, execution may terminate (eliminating the current interpretation), or may continue along one branch, or may split into several branches (exploring the tree depth-first). The behavior of interpretive

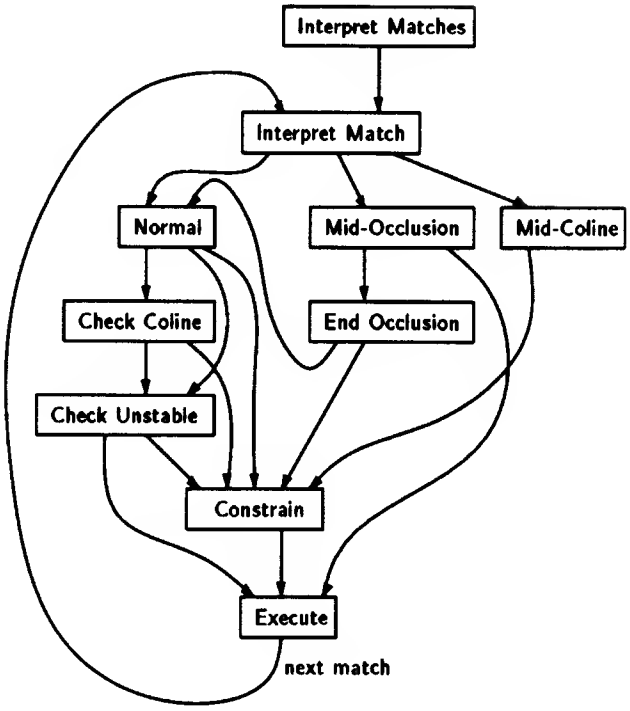

Figure 3.3: *A block diagram of the interpretation process at each node in the search tree.*

matching is therefore summarized by its behavior at a single node. Figure 3.3 is a block diagram of the interpretation process at a node. Figures 3.4 and 3.5 illustrate the flow of execution within each block, as described in the following sections.

3.2.2 Occlusion Interpretation

When object F (foreground) occludes object B (background), as demonstrated in Figure 3.6, we can expect that the image region caused by the surface of B will be bounded, in part, by the occluding edge of F. The occluding edge does not belong to B, or indicate its shape in any way. Therefore, when a correspondence is being hypothesized between the features in the group from B and the features in a model group, the features due to F should not be matched. Further, F obscures edges that do belong to B, so it may be correct to leave some model features unmatched to image features.

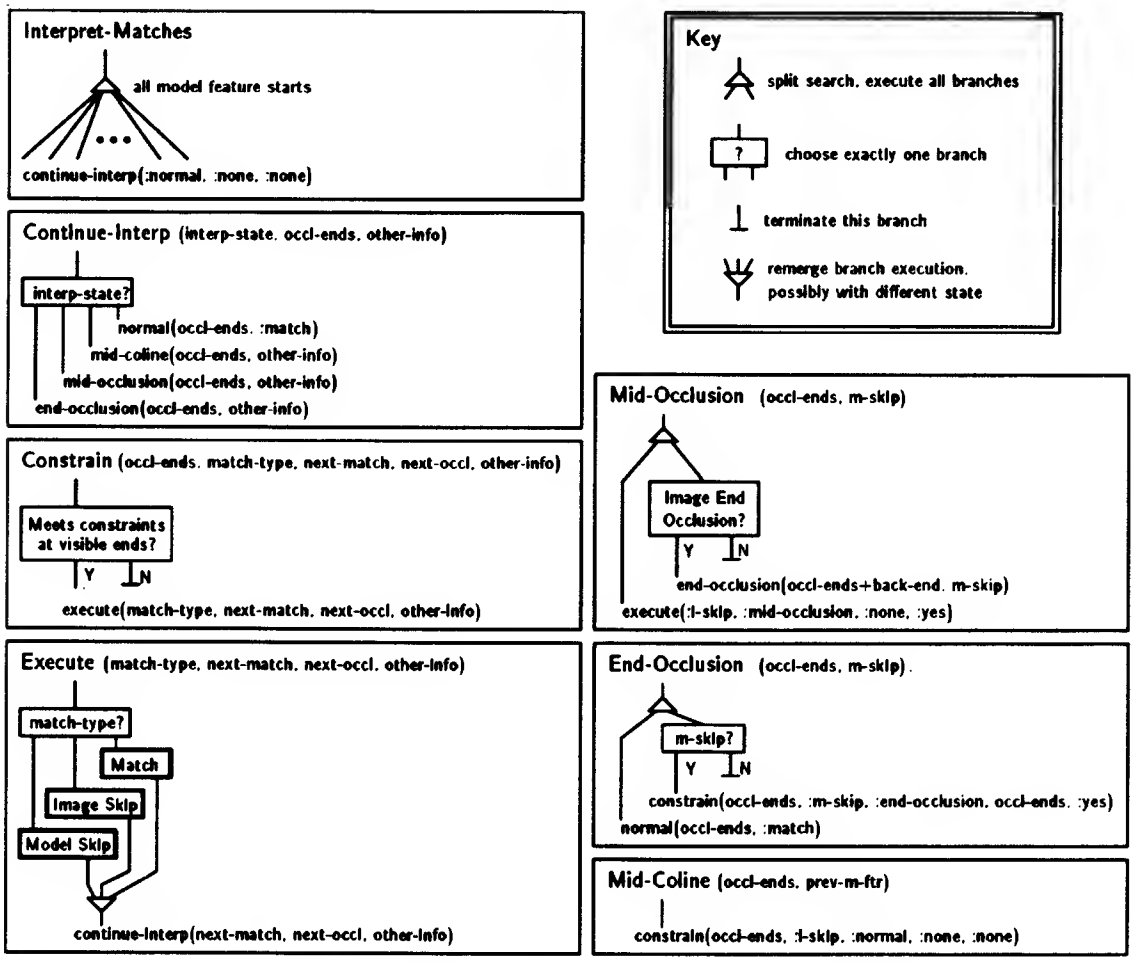

Figure 3.4: *The flow of execution through the interpretation modules. More modules are shown in the next figure.*

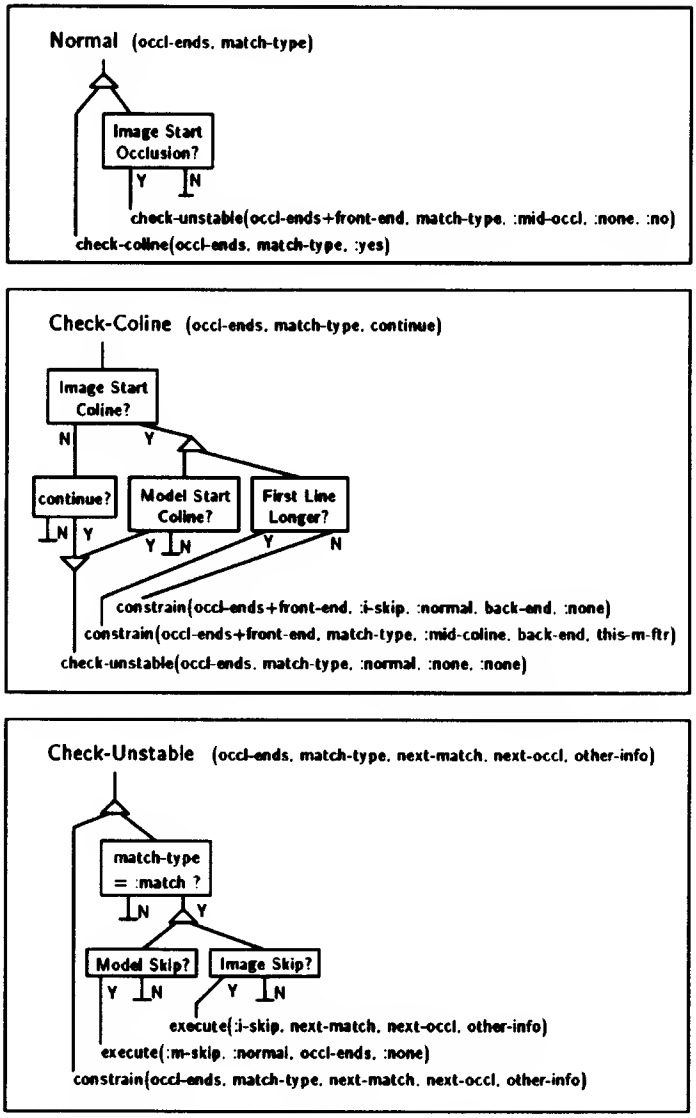

Figure 3.5: *The flow of execution through the interpretation modules, continued.*

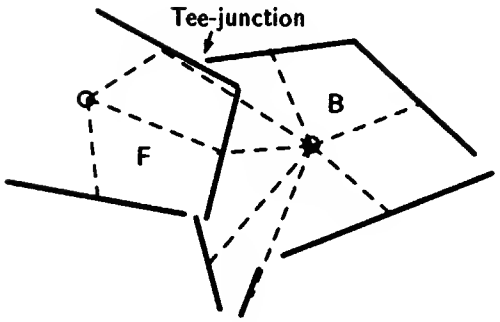

Figure 3.6: *An example of occlusion.*

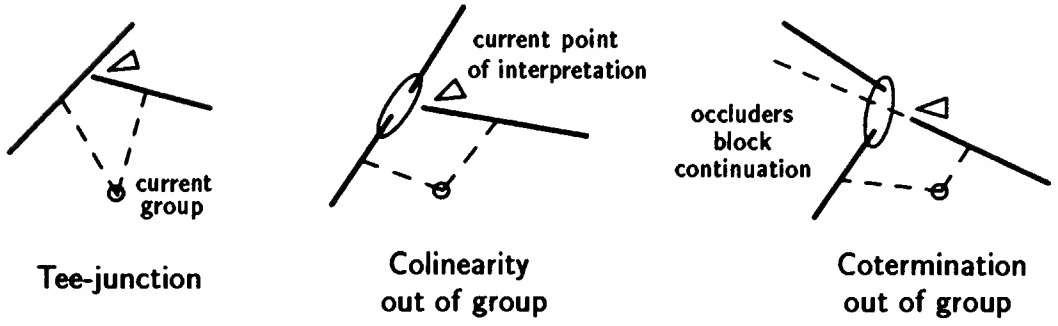

Figure 3.7: *Three configurations of lines and line-relations that are interpreted as occlusion.*

Forming Occlusion Plans from Feature Relations

The job of occlusion detection is to look for clues in the image that indicate when one object is in front of another. The Tee-junction has often been identified as the primary clue to occlusion, and a slightly generalized version of it is used in this module. From a grouping perspective, a feature is likely to be an occluder if it can be grouped with other features outside of the region group. Specifically, if it seems to continue outside of the region group, such as in a tee-junction, or via colinearity or cotermination with outside features, then occlusion should be hypothesized.

Figure 3.7 depicts these cases. The hollow triangle indicates the current point of interpretation. The question at each end of a line is, might this line have been terminated prematurely by an occluding object? If the nearby lines seem to be from an object other than the one forming the current group, as indicated by continuation beyond the group, and the object they belong to would cover the extension of the current line, then an occlusion hypothesis is formed. The tee junction fits this criterion, as does a colinear relation between the next line and a line outside of the group. In the third case, a cotermination between the next line and a line outside of the

group, an additional check is needed to determine if the occluding object blocks the extension of the current line. The current line splits the image into two half-planes. If some of the four endpoints of the two occluding lines lie in each of the half-planes, then they block the extension.

As the line features of a group are considered in turn, the occlusion condition is tested at each end. If the next features form one of the occluder cases, then the leading end is marked as an OCCLUSION-START in the plan. Symmetrically, if the previous features meet the occluder conditions, the trailing end is marked with an OCCLUSION-END. (Interpretation should always have clockwise/counterclockwise symmetry, since the direction in which the lines are considered around the group is arbitrary, even though their circular order is determined.)

Plans are also constructed for the model groups, using the same criteria, even though it does not make sense to say that occlusion is occurring in the model. It is important to detect occlusion conditions in the model, however, because they are likely to cause occlusion conditions in the image even when there is no occlusion. Without anticipating this situation, any tee-junction (for example) that happened to exist in the model would be incorrectly interpreted as an occlusion whenever it appeared in the image, unoccluded.

Using Occlusion Plans to Control Matching

During matching, there is always a current image feature and a current model feature under consideration. When the current image feature is marked with an OCCLUSION-START, an occlusion hypothesis begins. The occlusion hypothesis begins by creating a new branch in the match search. However, since the occlusion hypothesis is not certain, a normal match branch is also followed in addition to the occlusion branch. Under the occlusion hypothesis, the current image feature is matched, but subsequent image features are skipped until an OCCLUSION-END is found. Again, the end of occlusion is uncertain, so the occlusion hypothesis continues, and an additional branch is taken in which occlusion ends. When occlusion ends, it is not known how many model features were hidden. Both a match branch and a model-skip branch are generated. In the model-skip branch, at the next level, again a match and a model-skip are generated recursively. Thus, all model feature re-starts are tried. In Figure 3.4, this is implemented by the Normal, Mid-Occlusion, and End-Occlusion modules.

There is one possible alteration to this flow. If occlusion ends immediately after it begins, at the same location in the image, then it is safe to assume that no model features were hidden. This may occur when three or more lines coterminate at the proper angles. Then only the immediate model feature restart is tried. (The m-skip? variable controls this behavior.)

3.2.3 Feature Instability Detection

A very significant problem for interpretive matching is feature instability. If a model feature should be visible but is not detected in the image, then none of the features in the group will be successfully matched. Likewise, if an extra feature appears, it will disrupt the matches in the rest of the group. Additional instability comes from the grouping process. Some model group members are omitted in the image from certain viewpoints even though they are visible, and sometimes features that are not present in the model group are included in an image group. The goal of feature instability detection is to anticipate at least a few of these situations. Three independent strategies are employed.

Forming Instability Plans from Feature Relations

First, the most common cause of extraneous features in an image group are curved portions of an object. It would be very useful to have a smooth-curve-section feature, but this was not implemented. Instead, curved edges produce short image line features that do not correspond to any model feature. Therefore, line segments below a fixed length are marked UNSTABLE in the image plan.

Second, certain model features are known to be marginal members of certain groups, because they bound only a small portion of the region. These are marked UNSTABLE in the model plan. Unstable model group members might be detected automatically during model formation based on the images of the object in isolation, but generalization and greater care were needed so they were marked by hand.

Third, a common line feature instability occurs when a line is derived from a very slightly curving edge. The line may or may not be broken in the middle. Because of the prevalence of this phenomenon, coterminating lines that are almost colinear are treated specially. They are marked COLINEAR in the image plan. Also, roughly colinear model lines are marked COLINEAR, despite the fact that the model lines may be unambiguously distinct. As with occlusion, marking COLINEAR occurrences in the model allows a normal matching branch to be taken as well as the instability interpretation, when the COLINEAR conditions may be caused directly by the model rather than the line-breaking instability.

Using Instability Plans to Control Matching

When an UNSTABLE image feature is encountered, it generates two interpretation branches, a match and an image-skip. Likewise, an UNSTABLE model feature generates a match branch and a model-skip branch. (These mandatory branch splits can be very costly, in that they increase not only the number of branches explored, but

also the number of complete match sets that are found. Chapter 6 reports on this issue in greater detail.)

A COLINEAR image feature causes a match to be made between the model feature and the longer of the two colinear image features. This is accomplished by generating either an i-skip followed by a match at the next level, or a match followed by an i-skip at the next level, depending on the relative lengths of the line features.

3.2.4 Pose-Invariant Constraints

With the matching algorithm described so far, a match set would only be discarded if it prematurely ran out of either image or model features. Fortunately, there are geometric constraints that can be applied to prune the match tree much earlier. Their application depends on the current match set, but the constraints cannot depend on the model pose, since it is unknown during matching. As mentioned earlier, this leaves zero-length relations (cotermination, tee-junction) and zero-angle relations (colinearity, parallelism). Only these invariants are used, though other properties of the features may be strictly invariant under various conditions. For example, certain relations among more than two lines are strictly invariant if the lines are coplanar in the model, as shown by [Thompson and Mundy 87, Lamdan, Schwartz and Wolfson 87]. These also are not considered in the current implementation.

One additional geometric constraint is used, though it is not invariant. If two lines are nearly perpendicular in the model, then the lines are allowed to have any angle relation in the image *except* parallel. This is a very loose constraint, but it was necessary to restrict the otherwise unconstrained branching that occurred with image groups that happened to be mostly parallel, when matched to model groups without much cotermination. The constraint is not technically correct, since perpendicular model lines can appear parallel in the image when viewed from the side. However, groups that are mostly coplanar are only marginally visible from that perspective in any case.

Forming Constraint Plans from Feature Relations

The invariant relations correspond very closely to the elementary feature relations. To prepare for imposing constraint during matching, the features are simply marked with COTERMINATE, COLINEAR, TEE-INTO, and TEE-ACROSS, based directly on the basic relations. The only exceptions are the angle tests PARALLEL and PERPENDICULAR, which are tested between pairs of lines within each group. In the image group the starting feature is known, so the angle tests need only be applied with the previous lines as interpretation continues around the group. In the model, each starting position will be tried, so all pairs within each group are tested and

marked.

Using Constraint to Control Matching

The previous two interpretation modules were image-driven. The relations in the image were used to begin hypotheses and control branching, and to initiate comparisons with the model. Pose-invariant constraints are model-driven. If a constraint relation exists in the model, it must exist in the image. The converse is not true: any of the constraint relations can appear by coincidence in the image. The ability to impose these constraints so strictly comes from the fact that occlusion is explicitly hypothesized. If it were not, the absence of a constraint relation could always be blamed on occlusion. Under a particular occlusion hypothesis, certain model features are known to be visible, and others are known to be hidden. The hypothesis may be wrong, but the correct constraints may be strictly invoked in order to help determine its validity.

Many of the constraining relations exist between specific line endpoints, not just the lines in general. To apply these constraints, the correspondence not only of image and model lines, but of their endpoints, must be known. Fortunately, when region groups are formed, both the circular order of the features and each line's end-sense is determined.

Whenever two features are matched, the constraint relations of the current model feature with each previously matched model feature are found. If any model relations do not exist between the current image feature and the corresponding previous image features, the branch is terminated.

3.2.5 Interactions

Unfortunately, the interpretation plans described above are not independent. The order of their execution is important, and the combination of their resulting branches is not a simple union. Further, the state of each module can affect the operation and branches of other modules.

As just mentioned, the current occlusion hypothesis affects the application of constraints. This is incorporated by saving a list of occluded ends along with each feature match. Any constraint that applies to an occluded end is ignored.

Colinear instability interpretation also interacts with constraints. When two colinear image lines are matched to one model line, the model relations at one end are matched to one end of one image line, while those at the other model end are matched to one end of the other image line, even though the shorter image line is skipped. The unused ends of the image lines should be ignored for future constraints, and so are added to the occluded-ends list in the match.

The order of execution is a more entwined interaction. Some branches should only be taken if others are not. Some should be taken independently. The uneven combination of AND and OR conjunctions is evident in the flow diagram in Figure 3.4. Most of the modules have a "flow-through" branch which calls the next module in the standard sequence: Normal, Check-Occlusion, Check-Coline, Check-Unstable, Constrain, Execute. Diversions from this sequence occur when occlusion or instabilities begin. For example, when colinearity or occlusion begins, feature stability is circumvented and Constrain is called directly. Also, when an unstable feature is found, constraint is circumvented, since no match will be made. There are several more examples of this kind of interaction.

More significantly, the flow at the next level is affected by the NEXT-STATE argument to Execute, which is passed to Continue-Interpretation. This selects the entry point for the interpretation of the next match. Through this variable, the state of occlusion is maintained until an end is detected, for example. During occlusion, no instabilities are considered.

Several aspects of the matching algorithm are not represented in the flow diagram. The current image and model feature, as well as the other features in the groups, are passed through all modules, and updated by the Execute module. A termination test is included in the Execute module, and in the Continue-Interpretation module. Also, the value of BACK-END and FRONT-END is maintained, representing the trailing and leading ends of the current image line segment, respectively.

There is no inherent mechanism to prevent the various branches from duplicating. Duplication is minimized through the careful choice of branching in the modules, but sometimes it is unavoidable. In particular, two branches with the same match-type may be necessary if they have different state, such as different occluded ends or different hypotheses that will affect future levels of matching. It cannot be predicted whether these branches will diverge until they are explored. Once a match set is completed, only the correspondence of the features is important; the occluded ends and other hypotheses are not used for verification, though they have potential value there. After all branches have been explored, duplicate match sets are deleted using only the correspondence as a measure of equality.

3.3 Summary

In summary, the region groups create a very useful foundation for interpreting image phenomena and finding hypothetical feature correspondences that are consistent with the interpretations. They preserve the circular order of the lines in a group, and define the end-sense for each line. They also use an object's continuity to define its boundary and its inside, hypothetically, which allows occlusion to be detected.

In the image, occlusion is detected when the extension of a line is blocked, and feature instability is detected when a line is too short, or when it is colinear with another. In the model, group membership instability is anticipated for certain group members, and multiple groups are modeled to account for larger instabilities. These interpretations are performed before matching and are therefore match-independent. They are then used during matching to cause match branches to be taken only when deemed necessary. Match hypotheses are further constrained by several types of pose-invariant geometric properties of the model.

This approach is not exhaustive. It is a collection of cases, each reasoned from observed phenomena in the imaging process and from an understanding of image formation and feature extraction. It is expected that there are many other useful cases to incorporate, some that would improve reliability possibly at the cost of an increased number of match sets, and others that would improve efficiency possibly at the cost of reliability. Nonetheless, the current implementations of region grouping and interpretive matching narrow the search from trillions of possible matches to only a few hundred, as reported in Chapter 6.

Chapter 4

The Reggie System

Grouping and interpretive matching, described in Chapters 2 and 3, are two components of the complete recognition system, Reggie. In this chapter an overview of the system is presented, with a description of two other components: feature detection and verification. The remaining component, Model-Maker, is presented in the next chapter.

Figure 4.1 contains a block diagram of the Reggie System. It can be summarized as follows: a single grey-level image of a naturally lit scene forms the input. Edge pixels are found in this image using a standard edge detector, and the edge pixels are linked together into edge curves. The curves are smoothed and then approximated by a sequence of straight lines. The line segments are the only features used for recognition. As described in previous chapters, they are grouped by the open regions they bound, and elementary geometric relations are found among the lines.

Each image group is matched to each model group. Within the two groups, the correspondence between model and image lines is found by the interpretive matching module. The resulting match sets form the initial match hypotheses for verification. Zero, one or two model poses are found that align the projection of the model lines with the corresponding image lines, locally minimizing the sum of the squares of the alignment errors. For each pose, the rest of the model lines are projected into the image, and further matches are discovered. A match score is calculated, reflecting the similarity of the image to the projected model. Currently, all scores are reported, in an amount of time that does not depend on the number of instances. If a score is good enough, it may be interpreted as an instance of the model in the image.

Figure 4.2 shows the Reggie screen after finding the instance of the mouse model (upper right) in the image (upper left). The lower right-hand pane shows the match that led to the discovery: the image group, the model group, and the correspondence of the line features are shown. (There is no visible indication of the recognition pose in this screen, but it was correct.)

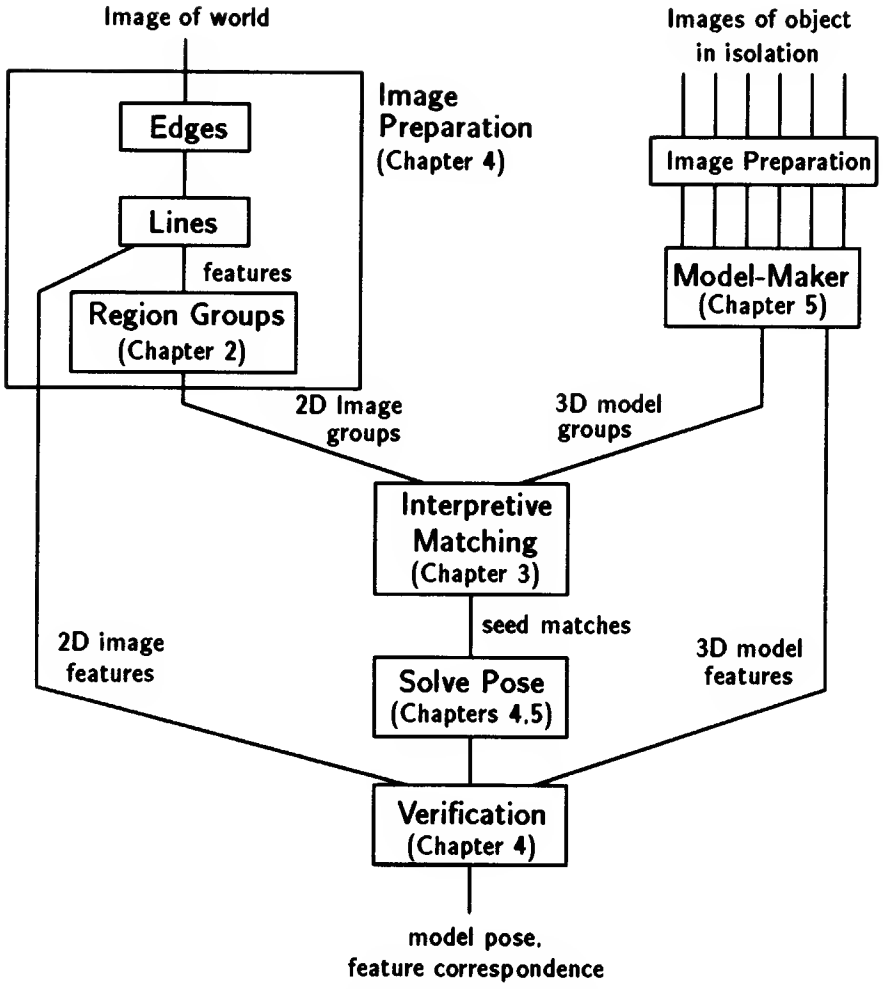

Figure 4.1: *A block diagram of the Reggie system.*

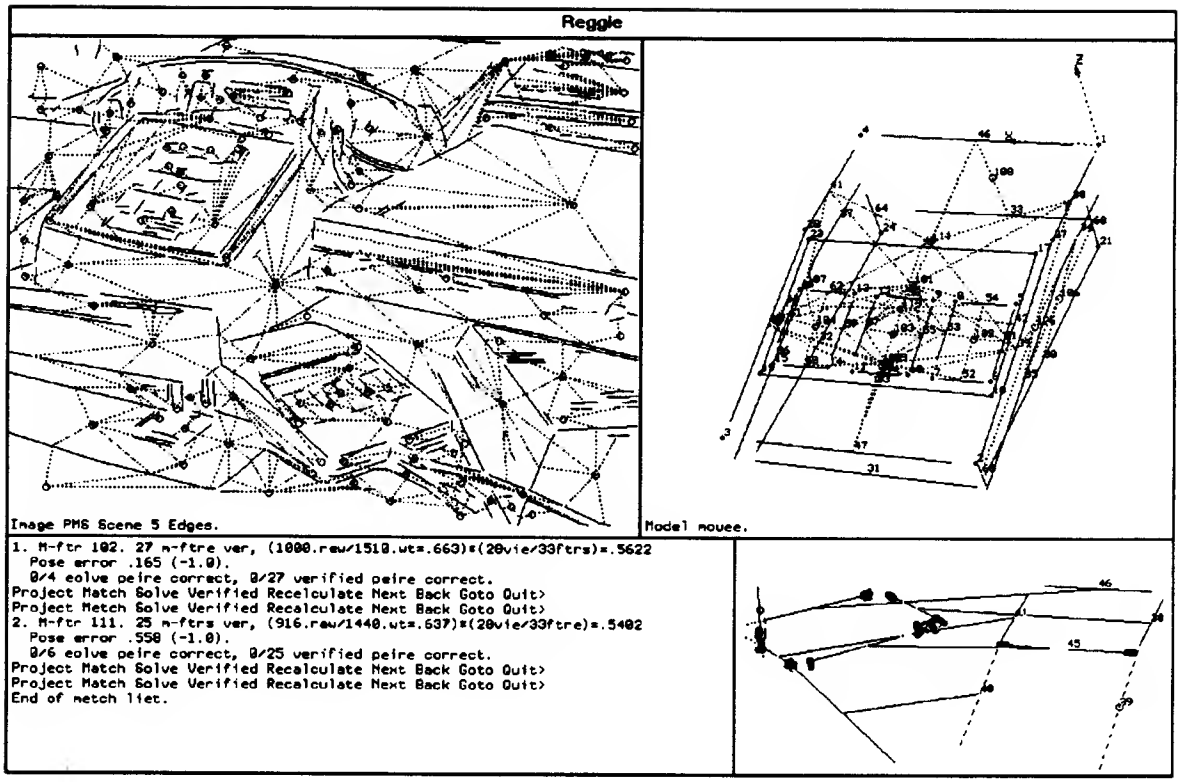

Figure 4.2: *An example of Reggie's screen. The mouse model was correctly discovered in the image, despite some occlusion.*

The next section describes the feature detection process in greater detail, and verification is described in Section 4.2.

4.1 Feature Detection

Feature detection consists of finding edges in the grey level image, connecting the edge pixels into curves, smoothing the curves, and finding lines to approximate the curves.

Edges

Edge pixels are found in the grey-level image using Canny's algorithm [Canny 86], with a Gaussian smoothing σ of 5 pixels. Reggie attempts to find edges and features at the finest scale possible, and then address the scale issue symbolically. The edge raster is scanned for edge-pixels. From each starting pixel found, a trace is initiated along an 8-connected path of edge pixels. As edges are found, they are removed from the image and collected into a curve. Since the starting point may be in the middle of a curve, the trace is initiated in both directions and the curves are joined. Where a curve intersects other curves and the trace would have to split, it is terminated.

The pixel curves are then smoothed only to remove effects due to spatial discretization. If only the pixel locations along a curve were needed, as in pose determination and verification, there would be no need for smoothing. However, since the curves are inspected to extract features, it is important to try to recover the underlying curve that caused the edge pixels. The purpose of features is to capture in a compact form the most salient information in an image. For Reggie, it is important that each feature represents edge pixels that are very likely to have come from the same object. The clues that support that interpretation are delicate, and benefit from sub-pixel accuracy. In particular, in order to find meaningful tangents and curvature at each pixel, information from neighboring pixel locations must be referenced and combined.

Smoothing is performed in two stages. First, the pixel squares are "dejagged". When a straight line is discretized into pixels, it forms a staircase pattern, as shown in Figure 4.3. Locally, the pixels can only represent horizontal, vertical, and diagonal directions. To see how the line may be recovered, imagine a string that is constrained to pass through the pixel squares, but not necessarily through their centers. If the string is pulled tight, it will form piecewise straight lines between the corner-contact points. The curve points are adjusted from the pixel centers to where they would fall on the string.

The second smoothing stage is more conventional. If each curve point were connected to its two neighbors by springs, and to its original (dejagged) location by a

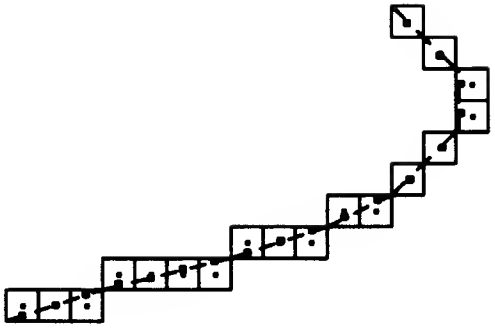

Figure 4.3: *Dejagging pixelated lines.*

spring, the whole curve would settle to a smoother shape. This is the result of the second stage.

Lines

Along the smoothed curves, straight sections are found. This is accomplished by approximating the entire curve with straight line segments, and then ignoring the short segments. This approach does not produce stable lines along smooth curves, but it succeeds in finding straight lines where they exist. First, each curve is closely approximated with a recursive split algorithm (courtesy of [Grimson and Lozano-Pérez 87], originally from [Horn 71]). It operates as follows: first, the entire curve is approximated by a single line. The curve is retraced, and the point is found where the distance between the line and the curve is maximized. If that error is above a threshold, the line is replaced by two lines, touching at the point of maximum error. Then, each of these lines is evaluated recursively. This algorithm is not guaranteed to find the minimum-error approximation, or the minimum-line approximation, but it does a fairly good job. One slight problem is that the lines are always on the inside of a circular section, rather than reducing their maximum error by being on both sides.

A more important problem is that a fairly straight section may be broken in the middle, as shown in Figure 4.4. On the first iteration, the largest error is in the middle, so the line is broken there. On the next iteration, each new line is broken again, near the points of high curvature. But now the two middle lines are nearly colinear. If they were merged, the error might still be below threshold. This problem is addressed by adding a stage to the end of the algorithm, which tests neighboring lines and merges them when possible. The merging stage may use a higher threshold than the splitting stage.

Not all line segments become line features. First, a line must be longer than 8 pixels. This test is intended to eliminate lines that come from smoothly curved

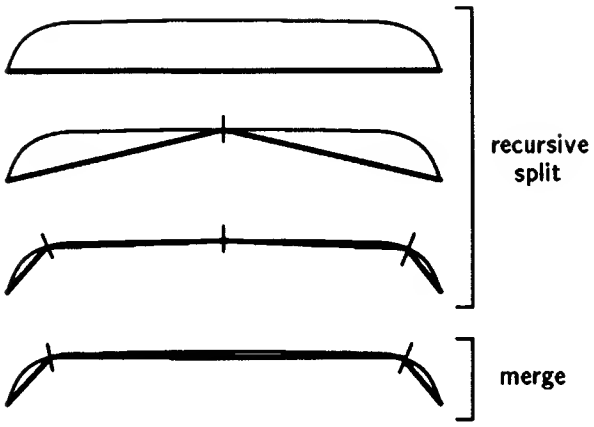

Figure 4.4: *Curves are broken into lines with a recursive split algorithm. A merge stage is added at the end.*

sections of the edge rather than repeatable line features of the model. Unfortunately, it also eliminates very short repeatable lines. To keep the length threshold as small as possible, the curvature of the underlying edge is also inspected. The curvature of an curve interval is measured as the change in tangent angle divided by the distance between the curve endpoints. At each end, the line segment is marked VALID if the curvature just beyond the end is more than twice the curvature along the interval of the line. If at least one end is valid, the line segment may become a line feature.

At this stage region-groups are found, as described in Chapter 2, followed by interpretive matching, as described in Chapter 3.

4.2 Verification

The result of interpretive matching is a collection of match sets. Each match set consists of at least three matches between an image line and a model line. Each is a hypothesis that needs to be verified or disproved using information from the entire model, instead of just one group. The task may be divided into two parts: solving for the pose, and scoring the projection of the model in the image.

4.2.1 Solving for the Model Pose (Exterior Orientation)

In order to project the model into the image, the pose of the model must be determined that best aligns the model lines with the matched image lines. The lines can be used directly for this purpose, but the solution is more reliable and easier to find if point correspondences are used instead. Therefore, vertices between lines are used whenever

possible. However, it is not correct simply to extend all matched lines and find the intersections. While any non-parallel lines will have an intersection in the 2D image, they must actually be coplanar in the 3D world in order to support the pose solution. This is very difficult to guess by looking at the image, but it can be easily determined in the model. Therefore, only image vertices between lines that are matched to model lines with a vertex can be used.

The problem of finding the pose of the model was confronted in the field of photogrammetry before it appeared in machine vision literature. In photogrammetry, images of the world are used to measure objects in it. If the sturcture of the world is known, then the orientation of the camera relative to the structure is often desired, based on the images and assuming perspective projection. This is called the *exterior orientation* of the camera, and is equivalent to finding the pose of the object relative to the camera. In [Bopp 78], a method is presented for finding the exterior orientation (and simultaneously calibrating the interior orientation of the camera) by iteratively finding the non-linearly constrained least-squares solution to a system of equations.

In vision, [Fischler and Bolles 81] describe a closed form solution for determining four solutions to the perspective exterior orientation problem, given three 2D image points matched to three 3D model points. The solution involves solving a quartic equation, and several other relatively expensive operations. Huttenlocher [Huttenlocher 88] recently developed a simpler closed form method for finding two poses that align three matched points, assuming parallel projection plus scaling. However, the neither method extends to lines, and neither can incorporate the valuable information available in the additional matches. Therefore, the Huttenlocher's analytic solution serves only as an initial pose guess for an iterative numerical solution, described below. The three points required by the analytic method are either the first three vertex matches in the match set, or the midpoints of line matches, if necessary.

From the initial guess, Newton's method is used to approach the solution iteratively, as done by Lowe [Lowe 87]. This method closely resembles the numeric solutions found by photogrammeters. Since the solution is overdetermined, the step is chosen that minimizes the least-squares error in the linearized problem. Newton's method requires the partial derivatives of the projected model points with respect to the pose. One of Lowe's contributions was to rediscover workable derivatives with respect to the three parameters of rotation. Using small angle approximations, he found very simple expressions that did not require extra rotation parameters. Another advantage of the method is that image error is minimized in a uniform way, giving even weight to every data point. Other methods had minimized 3D distances, which implicitly weighed the image data unevenly. Lowe also applied the method to lines. A detailed presentation is postponed until Chapter 5, where this approach is extended to solve for 3D models from several images of an object in isolation.

There is no guarantee that the iterative method will converge, or that it will

converge to the global minimum error. Much depends on the initial guess and the step size. Since there are two correct solutions to the exactly-constrained case (as shown by Huttenlocher), both are used as initial guesses for the over-constrained iterative solution. (The two initial guesses will be identical only in the special case that the three model points are in a plane that is parallel to the image plane.) Empirically, the two initial guesses can lead to two different pose solutions, or to the same solution. Sometimes one or both solutions diverge. However, the best local minimum found by the method has never yielded a pose that appeared incorrect. In all cases where the global minimum average error was less than ten pixels, the solution has converged very quickly to the correct solution. In cases where the match is incorrect, the pose solution is not important, and divergence actually increases efficiency since it prevents verification.

Every pose solution that fits the image with an average error less than ten pixels is then verified. (The error is the perpendicular distance from the projected model line endpoints to the matching image line.)

4.2.2 Completing and Scoring the Match Set

With the pose known, the model lines can be projected into the image. In the world, each line is only visible from a particular range of viewing directions and is hidden behind the object from other perspectives. If a model line would not be visible under the current pose then it should not be expected in the image during verification. To approximate visibility, each model line has a view-direction vector. If the dot product of the current viewing direction and the view-vector is positive then the line is included in verification. This is only an approximation because some lines may be visible from less than an entire hemisphere of viewing directions, and others may be visible from more viewing directions.

To verify the pose hypothesis, a local search is performed for similar image lines around each projected model line. In order to find nearby lines in the image efficiently, all image lines are indexed by image position. The image-index consists of an array that has an element for each four-by-four cell of the image. In each cell, a list is kept of the lines that occupy any pixel within a eight-by-eight square centered on the cell. By "quadruple-bucketing", all lines that are within two pixels of any given point are certain to be represented in the underlying cell, regardless of how close the point is to the edge of the cell. (This approach has been used by many researchers, possibly through reinvention, but an early documentation of its use is [Horn 72].) When a model line is projected into the image, the nearby cells are referenced to collect a list of potential matches.

The local image lines are evaluated by each model line. For each model line, a score and a weight are calculated. The scores and weights are summed over all visible

model lines, and the ratio of the total score to the total weight is the overall raw score.

First, all image lines are found that meet these criteria: the center of the image line must fall within 6 pixels of the model line, the angle difference must be less than 60 degrees, and the projection of the image line segment onto the model line must cover some portion of the model line segment. The combined coverage of the model line by all qualified image lines is then found. The score for the model line is the combined coverage, and the weight is its length in the image. In this way, model lines that are viewed nearly end-on (and thereby unstable) are weighed less heavily in the total score.

Since lines can break in unstable locations, two model lines may need to match to the same image line. However, an image line may not match more than two model lines. This is especially important to prevent high scores when a particular view of the model brings several model lines very close to each other in the image. The situation is most common when the model is small in the image and all distances are reduced. Under these conditions, the match should only be verified if there are also many image lines near each other.

The reverse is not restricted: a model line may find support from one, two, or more image lines when it has been interrupted in the image. The nature of the coverage measure prevents overlapping image lines from counting twice.

The raw score is multiplied by a visibility ratio: the number of visible lines divided by the total number of model lines. While it is true that views with fewer visible lines are more likely to be matched to chance alignments in the image, and are therefore less reliable, the visibility ratio unjustly reduces the chance of finding the model from certain poses. In particular, since the models do not contain any lines from the underside of the object, it cannot be recognized when it is upside-down. This may be interpreted as an a priori expectation for the object orientation, though that is not the original intention.

The matches are sorted by score and reported. Results for a set of images are presented in Chapter 6.

4.2.3 Future Work

Currently, Reggie can only find the best instance of the model, rather than all instances. This is entirely due to the weakness of the verification process. The earlier stages of the system often succeed in producing correct match hypotheses, independent of the number of instances, and the number of hypotheses that need to be verified is reasonably small. When a correct seed match is hypothesized, it usually has a higher score than any incorrect matches. However, the correct score is not usually enough larger than the best incorrect score that a threshold can be set. This

is especially true when the modeled object is more than half occluded.

The verification process could be improved in a number of ways. First, the pose could be readjusted after further matches have been discovered. Second, the model regions could be used to verify the absence of edges. This should greatly improve the discrimination of the score, since it operates over entire areas of the image, instead of just along lines. Third, the error tolerance could be adjusted for scale, and for distance from the seed matches. Currently, the error allowances are the same for all lines, and must be large enough to account for pose errors. When lines are close to the lines used to solve for the pose, they should be more likely to align in the image, and tighter tolerances could be used.

A more sophisticated improvement that might also have a large potential for increasing accuracy is occlusion interpretation during verification. If portions of the model are missing but have local justification for their absence, the score could be kept high. Missing lines without any evidence of occlusion could cause larger penalties. If the occluded section of the model were contiguous it would score more highly than disjoint occlusion evidence. Perhaps even a single consistent occlusion hypothesis could be formulated by finding grouping cues among the occluders.

Chapter 5

Model Building and Newton's Method

The three-dimensional models required by Reggie consist of lines with known 3D endpoints, relations of lines, and region groups of lines. This information could be generated manually by measuring the physical object and trying to anticipate which regions it might form in the image. When possible, however, it is better to automate the process, to save both work and guesswork.

Model-Maker is a sibling of Reggie. A sample of the Model-Maker screen is shown in Figure 5.1. It takes as input several images of the model object from different viewpoints (typically three to six). The camera viewpoints relative to the object are known only to within roughly 15 degrees. In each image, the Reggie line features and region groups are found. In addition, line intersection points (vertices) are computed. The vertices play a much larger role in model-building than in recognition, primarily because they are mathematically simpler. Manually, with the computer mouse, the correspondence of points among the images is indicated. With enough points, the three-dimensional location of the points can be solved, along with the precise relative pose of the camera for each image. This is known as the *relative orientation* of the cameras in photogrammetry. As in solving for the model pose (exterior orientation) in Reggie, Newton's method is used to iteratively find the solution that minimizes image error. From the point solutions and the image information, Model-Maker deduces solutions for the 3D lines, their relations, and the model regions.

One great advantage of using real images for model formation is that they provide a sample of the various image phenomena that will be encountered at recognition time. A CAD model would serve to predict the projected locations of the features, but for efficient, robust recognition, more knowledge is required. If certain lines are unstable, for example, this useful fact would be made apparent by its absence in some images. Most importantly, the appropriate groups can be extracted from the

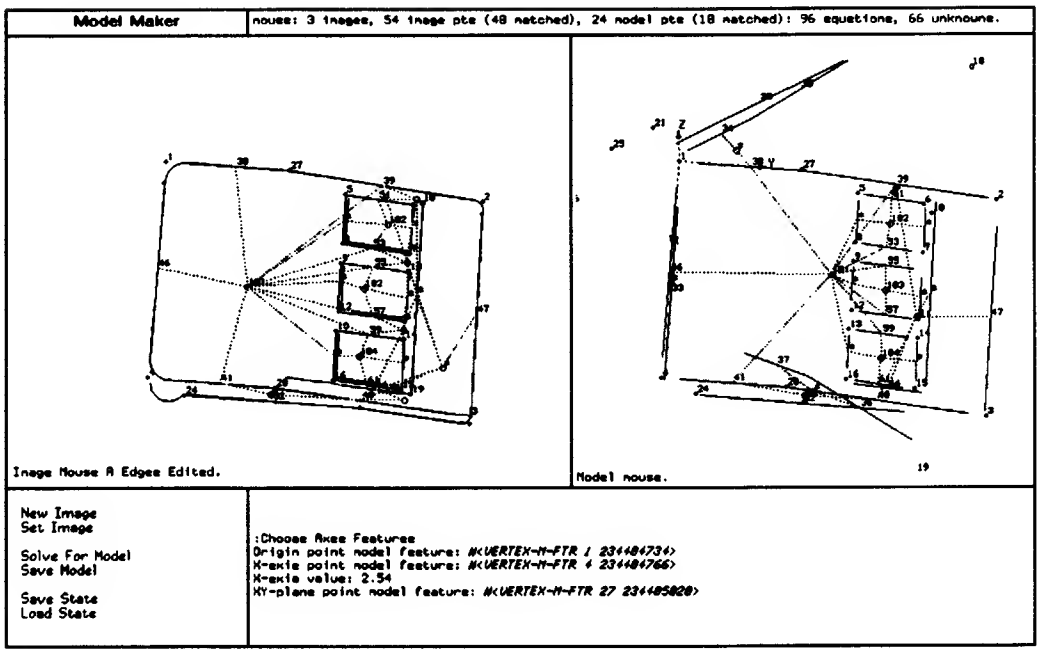

Figure 5.1: *A sample of the Model Maker screen. On the left is one of the images of the mouse in isolation. On the right is the incomplete mouse model. At this time, some of the model lines were visible in only one image, so their positions could not be determined. They retain the initial positions that were guessed from a single image.*

images. If the group-membership of a line is unstable, it can be marked unstable in that group. The natural images demonstrate properties of the object that would be hard to deduce with just a ruler.

Since model formation shares some of the difficult problems of recognition, it is not fully automatic. In addition to requiring point correspondences, Model-Maker also supports corrections and entry of supplemental information directly from the human operator.

In the Section 5.1, the manual and automatic operations supported by Model-Maker are outlined. In Section 5.2, the mathematics of the model solution are described in detail, including the solution of the pose from a single image as needed for recognition.

5.1 Manual and Automatic Operations

A large portion of the Model-Maker program is devoted to allowing the user to manipulate the model efficiently. Certain information is available in the images that must be identified by a human visual system, and then communicated to the program. Once this information is entered, the program can automatically solve for the precise three-dimensional structure of the model, and can infer some further information from the images.

The most significant human task is finding the correspondences of model points among the images. Without knowing the structure, it would be very difficult for the program to infer the correspondences among images from such different points of view. For a human, is is quite simple to identify corresponding locations on the model, though somewhat tedious. Model-Maker detects most image points automatically, and allows the user to specify the points with the mouse, and label the points by number. It shows the current model, such as it is, using the viewpoint from which the image was taken. When a new model point is identified, an initial 3D position is guessed for it. In the camera frame of reference, the x and y positions are taken from the image, and the z position is taken to be the distance to the model origin. If a model point was not automatically detected in an image, but should be visible, the user may create the point using the mouse.

Whenever there are enough matched points to solve for the model, the user may initiate that process. Those points that appear in at least two images will be determined, and others will retain their guessed initial positions. The mathematics of the solution are presented in section 5.2.

From the correspondence of the points, other correspondences may be inferred by the program. If both endpoints of a line have been matched, the line is automatically added to the model, either creating a new model line feature or matching to an

existing one. Likewise, when all the lines of a region group have been matched to the model, the region group is added. Currently, a new model region group will merge with an existing model region group if one is a subset of the other. It might also be useful to merge regions that have a large number of lines in common, and to mark the outlying lines as unstable, but this was not implemented.

Any line or region that is not automatically added to the model may be added manually, just as points are. Any feature may also be deleted from the model or from the image. (A list of deleted features is kept so that they may be recovered, if necessary.)

Instability information is entered by the user. Lines may be unstable features, in which case they do not always appear in an image, or their membership in a group may be unstable even though the line appears. A line may be marked as an unstable member of any particular group. Often, an unstable line feature will cause two entirely different groups to form, depending on its presence. In that case, both groups are independently kept in the model.

Finally, the line relations are entered by the user using the mouse. These are the cotermination, colinearity, and tee-junction relations that drive match constraints during recognition. These relations could probably be inferred from the images. Since they drive constraints, it would be conservative to require that, in order for a relation to be included in the model, it must appear in every image where it should be visible. But, since visibility and conservatism are somewhat subjective, it is left to the user to decide. (Parallelism is found automatically at matching time, since it is determined within each group, not among all pairs of lines.)

When the model is complete, it is saved, and can be used later for recognition by Reggie. While much of the work of building a model is still performed manually, it has proved to be worth the trouble. As discussed above, the multi-image approach reveals many of the imaging phenomena that will be encountered during recognition. Given that the model is to be derived from the images, Model Maker performs the indispensable task of solving for the image poses and model structure from the images. This would be unthinkably difficult to perform by hand. The next section explains this process in detail, beginning with a general review of Newton's Method.

5.2 Solving for Camera Poses and Points

Newton's method is a general iterative approach to solving a system of non-linear equations, represented as functions set equal to zero. It requires the first derivatives of the functions with respect to the solution variables (as functions of the solution variables). The derivatives are used to calculate the variable values that would bring the functions to zero, if they were linear. If the system of equations is overdetermined,

the functions cannot necessarily be brought to zero simultaneously. In that case, the well known pseudo-inverse method is applied to find the variable values that would minimize the sum of the squares of the functions, if they were linear. The new variable values are the next guess. Since the functions are not linear, the new values may not be zero (or least-squares minima). The process is repeated until the functions are as close to zero as desired.

Lowe applied Newton's method to the problem of finding the pose of a model, such that its projected features align with the corresponding image features. (This is very similar to the photogrammeters solution to the relative orientation problem, discovered earlier.) In this application, the functions are the projection equations (minus the image data), and the variables are the six degrees of freedom of the pose. Lowe derived the equations for points and lines, using simple derivatives of the image error with respect to the three inherent rotation variables. This is the method used by Reggie to solve for the pose.

This standard method also forms the basis for the model solution. The extension of the method required determining the derivatives of the image error with respect to the three-dimensional positions of the model points. In Model-Maker, the image points from all the images are combined in a large system of simultaneous equations. The solution provides the camera poses and the 3D positions of the points. With the camera poses known, the 3D locations of the lines in the images can be solved independently. This application of Newton's method goes beyond Lowe's work, since he determined how to solve for a pose from image and model lines, while Model-Maker solves for model lines given image lines and poses.

5.2.1 Newton's Method

Since Newton's method is applied in so many cases, I developed a generic program to perform Newton's method on any Lisp object that can provide the essential information about a system of equations:

$$\begin{aligned} f_1(x_1, x_2, \ldots x_n) &= 0 \\ f_2(x_1, x_2, \ldots x_n) &= 0 \\ &\vdots \\ f_m(x_1, x_2, \ldots x_n) &= 0. \end{aligned}$$

The Lisp object contains the solution variables, X. Since the functions are supposed to be zero, their values are collectively called the error. The object must be

able to fill a vector with the current error, F. It must also be able to fill a partials matrix (Jacobian) with $\frac{\partial f_i}{\partial x_j}$ at the current value of X. Finally, it must be able to update X given Δ_X. Each of these operations are called "methods" of the object. The job of the generic program is to drive the iteration, and on each iteration to call the object methods, use the partials matrix and error to solve for Δ_X, and check for convergence.

The process begins with an initial guess at the solution, X_0. The object must provide this guess in any manner appropriate to its problem. If there are multiple solutions, the initial guess will affect which solution is found by the iterations. It may also affect the convergence of the iterations, and the number of iterations required to reach the solution.

The Single Variable Case

The iterative procedure is demonstrated graphically in Figure 5.2. At iteration n, the solution value is $x_{[n]}$, and the error is $f(x_{[n]})$, also denoted $f_{[n]}$. Using the value of $\frac{\partial f}{\partial x}(x_{[n]})$, the tangent to $f(x)$ is extended until it crosses the $f = 0$ line. The value of x at that point is taken as $x_{[n+1]}$, the solution variable value on the next iteration. If $f(x)$ was actually a linear function of x, this is where it would be zero. Since it is not necessarily linear, it will not necessarily be zero at the new guess. However, depending on the function and the initial guess, the new $x_{[n+1]}$ will be closer to x^*, the true solution. As the iteration proceeds, if it converges, the value of x will become arbitrarily close to x^*. In practice, if the function is fairly smooth, convergence will be rapid.

In this example, there is only one x and one f. This makes solving for $x_{[n+1]}$ straightforward. If f were a linear function, then

$$f_{[n+1]} = f_{[n]} + \left.\frac{\partial f}{\partial x}\right|_{x_{[n]}} (x_{[n+1]} - x_{[n]}).$$

Since we are solving for $f_{[n+1]} = 0$,

$$x_{[n+1]} - x_{[n]} = -\left(\frac{\partial f}{\partial x}\right)^{-1} f_{[n]},$$

where the partial derivative is implicitly evaluated at $x_{[n]}$. From this, with one x and one f, we can easily solve for $x_{[n+1]}$.

The Multi-Variable Case

When there are more solution variables and more error functions, the equation becomes

$$-J\Delta_X = F,$$

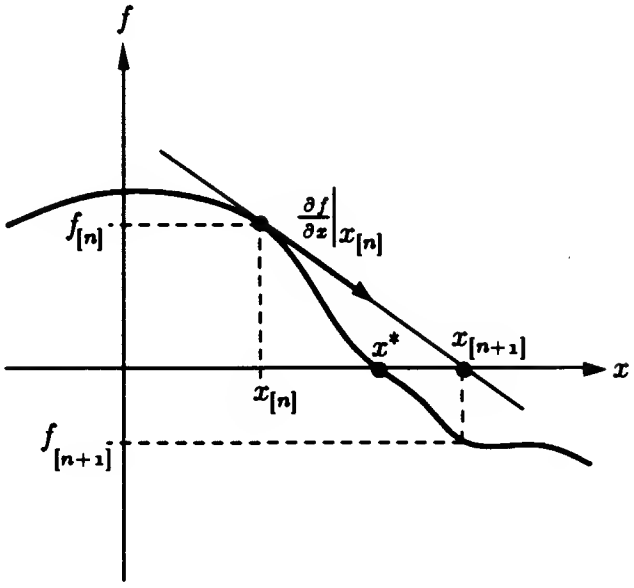

Figure 5.2: *A single iteration in Newton's method for solving one equation in one variable.*

and the solution is

$$\Delta_X = -J^{-1}F,$$

where

$$\begin{aligned} J &= \left[\frac{\partial f_i}{\partial x_j}\right]\Big|_{X_{[n]}} \\ \Delta_X &= X_{[n+1]} - X_{[n]} \\ F &= F(X_{[n]}). \end{aligned}$$

J is the Jacobian matrix, and X and F are vectors of x_j and f_i, respectively. Subscripts with square brackets indicate the iteration number, while unadorned subscripts distinguish different functions and variables.

The iteration is now harder to visualize because it appears in a space that has more than three dimensions. Figure 5.3 demonstrates the iteration procedure for just two variables, x_1 and x_2, and two functions, $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$. The four-dimensional space of x_1, x_2, f_1, and f_2 is shown as two three-dimensional spaces. At the beginning of an iteration, at $X_{[n]}$, there is a value for each function f_1 and f_2 (labeled "tangent point" in each space). First consider f_1. The partials of f_1 with respect to $X = (x_1, x_2)$ define the tangent plane at that point. The tangent plane

extends through the $f_1 = 0$ plane, forming a whole line of possible X values where f_1 would be zero if it were a planar function. Likewise, the tangent plane for f_2 intersects the $f_2 = 0$ plane along a line in the X plane. The intersection of these two lines is where both error functions f_1 and f_2 would be zero, if they were planer. This is taken as $X_{[n+1]}$, the next solution guess. However, the actual functions f_1 and f_2 intersect the $f = 0$ plane along two curves. At the intersection of the curves, X^*, lies the true solution to the non-linear problem. If the functions are smooth enough and slope towards the solution, $X_{[n]}$ will move closer to X^* as the iterations progress.

The Overdetermined Case

In both of the above examples, there are as many functions (equations) as variables. When the problem is overconstrained, as it usually is for the applications in Reggie and Model-Maker, the method must be modified. There will no longer be a value X^* where all the functions are zero. Instead, X^* will denote the value of X that minimizes the sum of the squares of the error functions. On each iteration, $X_{[n+1]}$ will minimize the sum-squared error as if the functions were linear about $X_{[n]}$.

A single iteration for the case of two functions and one variable is illustrated in Figure 5.4. The three-dimensional space contains a curve which has an f_1 and f_2 value for every x. To the right, the curve is projected into the f_1, f_2 plane, where the x-axis is not visible, but x parameterizes the curve. In the F plane, the distance from the origin is equal to (the square-root of) the sum of the squares of the error function values, which is the value that is to be minimized. As before, the partials of the functions with respect to x define the tangent at $x_{[n]}$. If both f_1 and f_2 were linear, the least-squares solution would be where the distance between the origin and the tangent line is minimized. This occurs where the line between the origin and the tangent is perpendicular to the tangent. From this location, the change in x is found. Since the curve is not linear, and since a unit step in x does not necessarily correspond to a unit arclength step along the curve, the actual value of F at $x_{[n+1]}$ will not be minimized, and so the iterations continue. They may converge towards x^*, where the curve is closest to the origin: the true least-squares solution.

Mathematically, the solution at each step is found by solving an equation similar to the exactly-determined case. Previously, a solution of

$$J\Delta_X + F_{[n]} = F_{[n+1]} = 0$$

was found. But since J is no longer square, it cannot be inverted. We cannot, in general, find a value for Δ_X that makes $F_{[n+1]}$ equal to 0. Instead, we wish to minimize the sum of the squares of $F_{[n+1]}$:

$$(J\Delta_X + F)^T(J\Delta_X + F) = \Delta_X^T J^T J\Delta_X + F^T J\Delta_X + \Delta_X^T J^T F + F^T F.$$

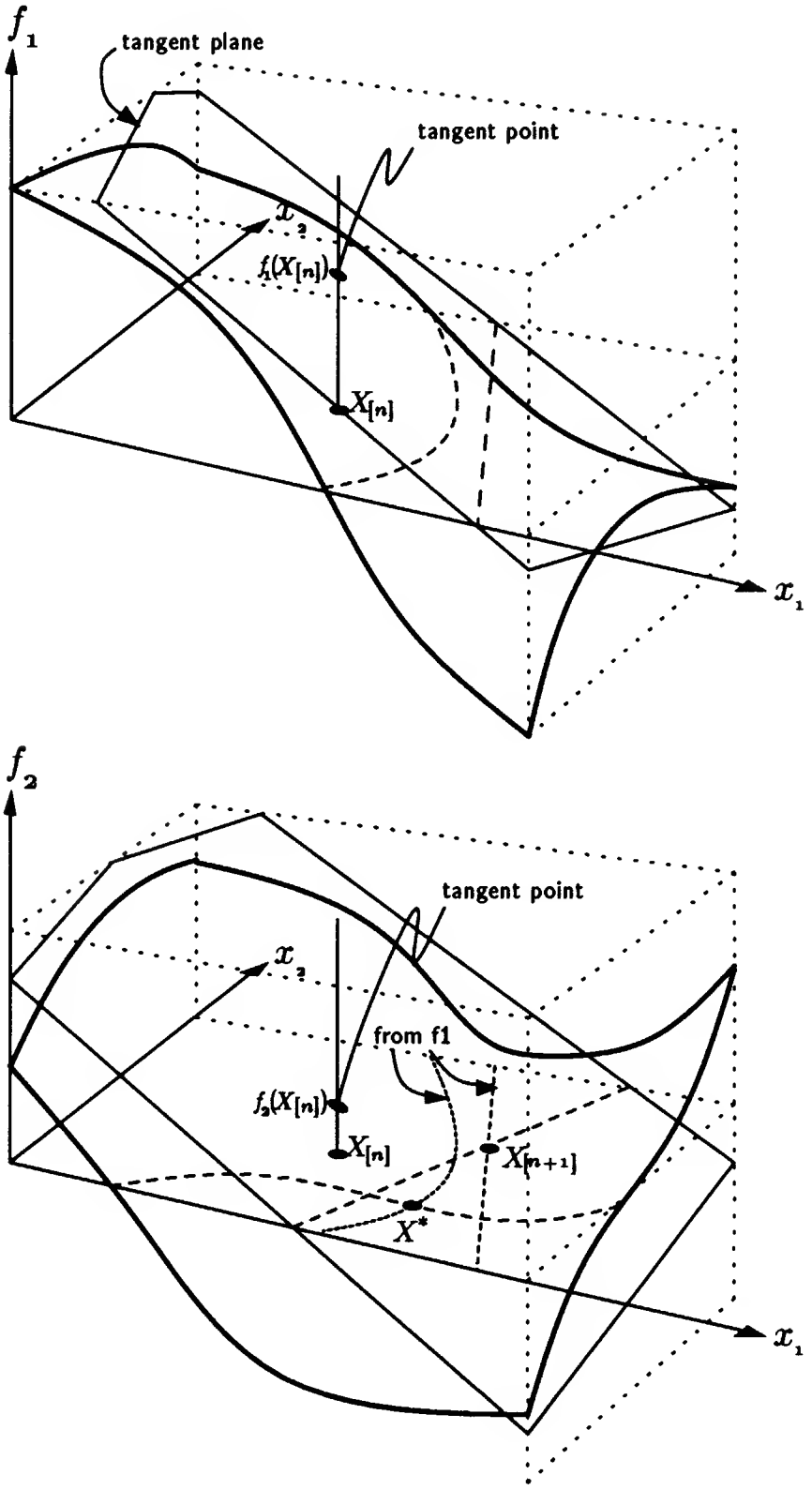

Figure 5.3: *A single iteration in Newton's method for finding the simultaneous zeroes of two functions in two variables. The two graphs are the two functions, they are not "before" and "after".*

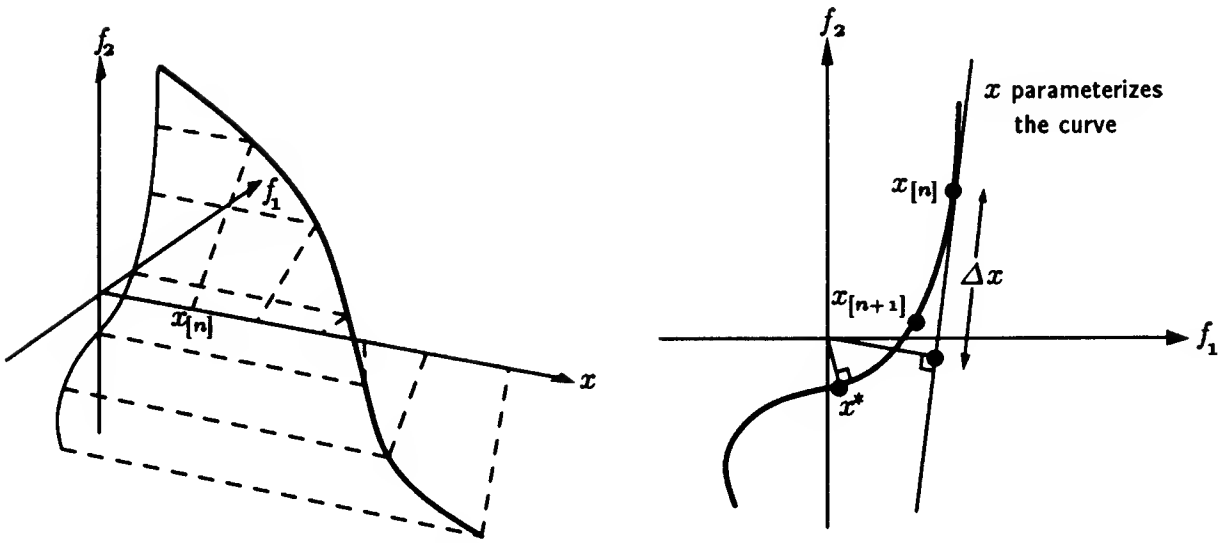

Figure 5.4: *A single iteration in Newton's method for solving two overdetermined equations.*

To minimize this expression over Δ_X, its derivative (gradient vector) with respect to Δ_X is set to the zero vector, yielding

$$(J^T J)\Delta_X = -J^T F,$$

or

$$\Delta_X = -(J^T J)^{-1} J^T F.$$

$(J^T J)^{-1} J^T$ is called the pseudo-inverse of J, and this method of solving overdetermined equations is known as the Gauss Least-Squares method.

Convergence

From the illustrations of the iteration process, we can see that convergence is a difficult property to ensure. It depends on the starting guess, but also on the specific shape of the error functions. Consider the overdetermined system in Figure 5.5. The functions are very far from linear around x^*. Assume that x parameterizes the curve such that a unit step in x corresponds to a unit step in arclength along the curve. At iteration n, the value of $x_{[n]}$ and the tangent indicate that a large step in x should be taken, and it will bring the solution very close to zero. In fact, when the step is taken, at $x_{[n+1]}$ the error has already passed through its true minimum at x^* and continued out to a larger distance from the origin. It is possible that the iterations will diverge from x^*.

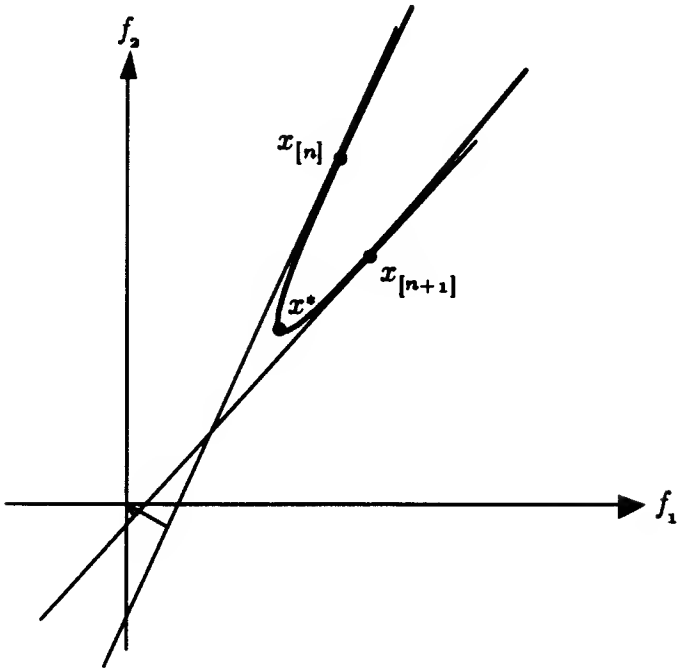

Figure 5.5: *In this case, Newton's method might diverge from the solution.*

In practice, divergence did occur in the some of the applications. The problem is avoided in two ways. First, the Δ_X step can be scaled, bringing the solution closer, but reducing the likelihood of overshooting X^*. Second, convergence can be forced. Without knowing X^*, it is impossible to know how to get closer. But it is possible to require that the sum-squared error always decreases. If a step causes the error to increase, it is scaled down. This process is repeated until the error decreases. If the error functions are continuous, it must be the case that some step size will decrease the error, as the derivatives indicate.

Both methods are available in the generic Newton solver, and are employed by most of the applications.

5.2.2 Image and Model Points Known, Find Pose (Exterior Orientation)

The first application of Newton's method is to solve for the model pose, given the image and model, and the correspondence of their points. The solution variables, X, are the rotation angles and translation vector of the pose. The error functions, F, are the x and y distances between the image points and the projected model points. Let $\vec{x_m}$ be a three dimensional model point. First the point is rotated, then translated.

Let

$$\vec{x_r} = \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = R\vec{x_m}$$

be the rotated model point, and

$$\vec{x_{rt}} = \begin{bmatrix} x_{rt} \\ y_{rt} \\ z_{rt} \end{bmatrix} = \vec{x_r} - \vec{t}$$

be the rotated-translated model point. Finally, the point is projected into the image,

$$\vec{x_{mi}} = \begin{bmatrix} x_{mi} \\ y_{mi} \end{bmatrix} = \begin{bmatrix} c(x_r - t_x) + x_{cp} \\ c(y_r - t_y) + y_{cp} \end{bmatrix},$$

where $c = \frac{f}{t_z - z_r}$, and f, x_{cp}, and y_{cp} are the focal length and center-of-projection of the camera. The resulting coordinate frames are shown in Figure 5.6.

The error functions are then

$$\begin{aligned} f_x &= x_{mi} - x_i \\ f_y &= y_{mi} - y_i \end{aligned}$$

for each of the measured two-dimensional image points. (i denotes an image coordinate, not the ith point.) Conveniently, $(x_{mi} - x_i)^2 + (y_{mi} - y_i)^2 = d^2$, the square of the distance between the image and projected model points. This quantity is rotationally symmetric, and arguably attractive to minimize.

To apply Newton's method, the partials of the error functions with respect to $\vec{t}$ and the three rotation variables are required. The above equations are fairly straightforward to differentiate, but they hide the fact that the rotation angles are buried in transcendental functions in the R matrix. Even when other representations of rotation are used, such as quaternions, the derivatives are relatively expensive to compute. To avoid this complication, as Lowe point out, the rotation can be represented as a large rotation, R, compounded with incremental rotations about the rotated axes, ϕ_x, ϕ_y and ϕ_z. The derivatives with respect to the incremental rotations are quite compact:

	x_r	y_r	z_r	c
$\frac{\partial}{\partial \phi_x}$	0	$-z_r$	y_r	$c^2 y_r$
$\frac{\partial}{\partial \phi_y}$	z_r	0	$-x_r$	$-c^2 x_r$
$\frac{\partial}{\partial \phi_z}$	$-y_r$	x_r	0	0

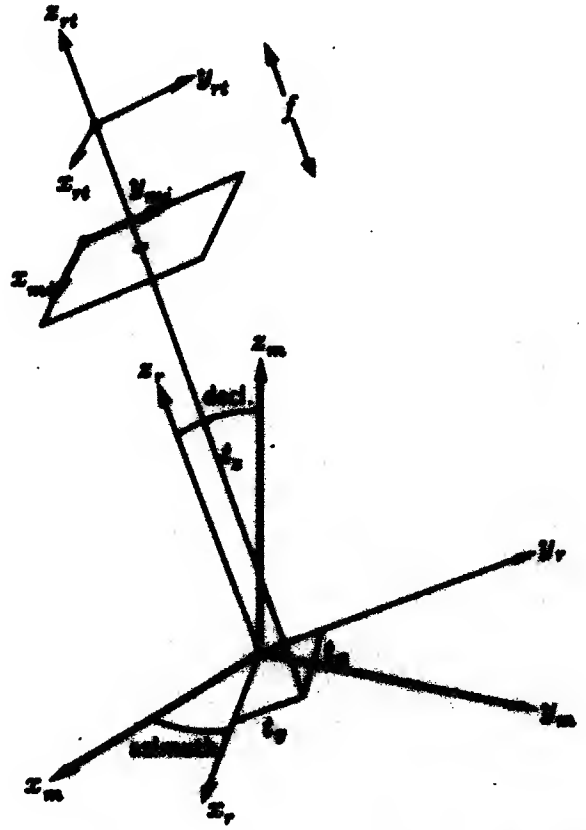

Figure 5.6: *The coordinate frames used for rotating, translating, and projecting a model point into the image.*

Using these, the Jacobian (shown transposed) is

	x_{mi}	y_{mi}
$\frac{\partial}{\partial t_x}$	$-fc$	0
$\frac{\partial}{\partial t_y}$	0	$-fc$
$\frac{\partial}{\partial t_z}$	$-fc^2x_{rt}$	$-fc^2y_{rt}$
$\frac{\partial}{\partial \phi_x}$	$fc^2y_rx_{rt}$	$fc^2y_ry_{rt} - fcz_r$
$\frac{\partial}{\partial \phi_y}$	$fc^2x_rx_{rt} + fcz_r$	$fc^2x_ry_{rt}$
$\frac{\partial}{\partial \phi_z}$	$-fcy_r$	fcx_r

The changes in ϕ_x, ϕ_y and ϕ_z prescribed on each iteration are converted to single-axis rotation matrices, which are applied sequentially to the pose rotation matrix, R. This relies on the fact that small rotations are almost commutative. The translations are simply added directly.

With six variables and two equations per matched point, at least three matched points are required. As described in Chapter 4, an analytic method is used to form the initial guess from just three points (assuming parallel projection plus scale). When the match is correct, Newton's method iterates rapidly to the least-squares solution for all the matched points (using true perspective).

However, especially when even the best pose has a large image error or when the image features are very close together, the solution can diverge. In the first case, large steps in angle may be prescribed. Since the rotation partials and update procedure are derived under small-angle assumptions, the solution can jump out of the convergence region. To reduce this effect, angle steps are truncated at 45°. In the second case, the proximity of the image features may mean that the error is minimized when the object is infinitely far away. The model features then all project to a point in the middle of the image features, such that offset penalties are constant and orientation penalties may be reduced independently. In this case, however, the matrix becomes singular. To prevent this, the pose z-translation is forced to stay in a fixed range, such that the object cannot appear larger than the entire image, nor can it appear smaller than roughly twenty pixels across. This also prevents solutions based on inverted versions of the model that occur when the object is behind the camera.

5.2.3 Image and Model Lines Known, Find Pose

Lowe also derived the equations to solve for the pose from correspondences of line features between the model and image. If line endpoints were reliable, they could

simply be used as points. However, while the orientation and offset of a line feature are usually quite stable (proportional to its length), the exact point of termination along the line is very unstable, especially due to occlusion. Therefore, only the orientation and offset of the image line are used to solve for the pose. This is accomplished by minimizing the perpendicular distances of the projected model line endpoints to the image line. Note that the model line endpoints are known—there is no need to avoid them and use only the model line direction and offset. However, if only one line were involved in the solution, a large t_z would bring the model points closer together, thereby reducing the distance to the image line without imposing the rotation restriction. This is avoided only because other lines impose their offsets, requiring sufficient separation of points in the image.

An image line may be represented by the equation

$$n_x x + n_y y = d,$$

where n_x and n_y are components of the unit normal vector of the line, and d is the offset. For a model line endpoint projected into the image, the error function is the perpendicular distance from the point to the line;

$$f = n_x x_{mi} + n_y y_{mi} - d.$$

A single error function is produced by each model endpoint. Note that the function is a linear combination of the error functions for points, plus a constant. The partials are therefore simply linear combinations of the point partials shown above. For any given pose solution variable, q,

$$\frac{\partial f}{\partial q} = n_x \frac{\partial x_{mi}}{\partial q} + n_y \frac{\partial y_{mi}}{\partial q}.$$

5.2.4 Image Points Known, Find Poses and Model Points (Relative Orientation)

The previous two applications were part of recognition, where the structure of the model is known but its pose is not. When building a model, however, the 3D structure must also be deduced. This cannot be done from a single image—it requires at least two, and may need several more to include features from all sides of the object. Each new image introduces six unknown pose parameters. Since the pose solutions depend on the points, and the point solutions depend on the poses, they must all be solved together as a large system of equations.

For two images, [Horn 87] proposes a new and simpler iterative least-squares scheme that minimizes the perpendicular distance between the rays extending out

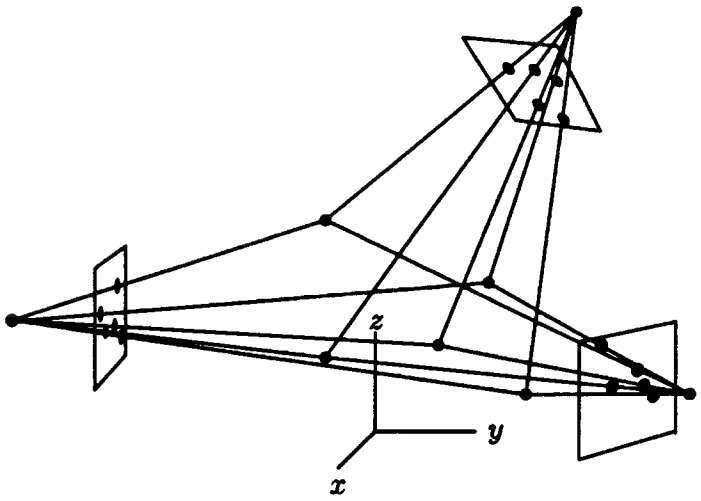

Figure 5.7: *Three cameras viewing five points in space. Regardless of how many images and points are available, the scale, rotation and translation of the points and cameras as a whole cannot be determined from the images.*

from each camera's center of projection through the image points. His method could probably be extended to apply to more images, but the methods already developed for Reggie extend uniformly to the various problems, and they minimize the error where it is measured: in the image.

In recognition, the pose is usually thought of as the pose of the model relative to the camera, in the real world. With multiple images, it is more convenient to fix a coordinate system on the model and think of each pose as the rotation and translation of the camera relative to the model. The equations are the same: the error functions still come from points in the images, and the pose variables still appear in the same way.

However, since the 3D points are now variables, too, some further partial derivatives are needed. Unlike the earlier rotation derivatives, they make direct reference to the elements of the R matrix:

	x_{mi}	y_{mi}
$\frac{\partial}{\partial x_m}$	$fcr_{00} + x_{rt}fc^2r_{20}$	$fcr_{10} + y_{rt}fc^2r_{20}$
$\frac{\partial}{\partial y_m}$	$fcr_{01} + x_{rt}fc^2r_{21}$	$fcr_{11} + y_{rt}fc^2r_{21}$
$\frac{\partial}{\partial z_m}$	$fcr_{02} + x_{rt}fc^2r_{22}$	$fcr_{12} + y_{rt}fc^2r_{22}$

Another new consideration is that some variables are not observable from the image data. Imagine three cameras viewing five points in space, as shown in Figure

5.7. Each camera is a projection point with a square image plane in front of it. The distance from the projection point to the plane is the focal length. Rays extend from the camera projection points to the five model points, passing through the image squares. Where they pass through, they form image points. Now consider if all three cameras move back twice as far (but maintain their focal lengths and image size), and the model points expand to make all their mutual distances twice as large. The images will be exactly as they were before. Since the images are the only evidence we have of the world, there is no way to solve for the overall scale of the points and camera distances. Similarly, imagine an absolute reference frame in the space. Everything can rotate and translate rigidly with respect to the absolute frame without altering the images. So, there are a total of seven parameters that cannot be found: the absolute 3D translation, 3D rotation, and scale. These may be set arbitrarily, but they should not be part of the solution variables.

Consequently, the entire pose of the first image is not included, and the z-translation of the second image pose is not included. These values may be set arbitrarily, so they are left at their initial estimates. However, for convenience, an absolute coordinate system on the model is specified by the user, by choosing one model point to be the origin, a second point to be along the x-axis, and a third to lie in the x-y plane. Further, the absolute scale is specified by giving the x-value of the second point. After the solution has been found, all points and poses, including the "fixed" poses, are scaled and rigidly rotated to incorporate the chosen absolute coordinates.

With the seven variables omitted, the combined partials matrix is laid out as in Figure 5.8. The black rectangles indicate non-zero blocks. Each row is an equation, with two equations per image point. Each column is a solution variable, with six columns per new image (except the second, which has only five), and three columns per distinct 3D point. Each image has a block of rows, with point entries depending on which points appear in the image. The total number of equations must be greater than or equal to the number of variables. There are $6I - 7 + 3P$ variables, where I is the number of images, and P is the number of distinct 3D points. There are $2\sum_{j=1}^{I} p_j$ equations, where p_j is the number of image points visible in image j. Not all images will contain all points, but every 3D point must appear in at least two images.

Empirically, with reasonable initial camera pose guesses, the solution converges in ten to twenty steps. Each step is slow, however, because the pseudo-inverse involves multiplying the Jacobian by its transpose. The time required for matrix multiplication is proportional to the cube of the dimension of a side, which is in the neighborhood of one hundred. When convergence is forced, however, the pseudo-inverse only needs to be calculated once per step, and then the appropriate size for the step can be found by trial and error.

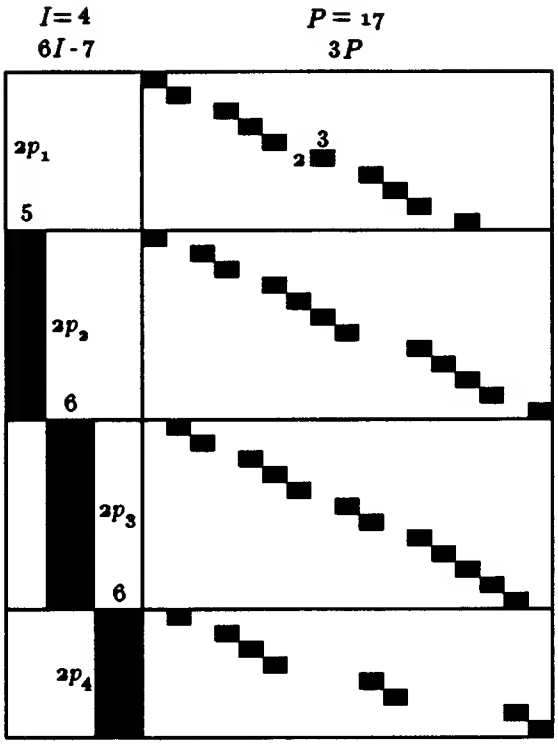

Figure 5.8: *The partials matrix layout to solve for the camera poses and model points using Newton's method. The white areas are zeros. Note that there is no block for the first pose, and the second pose block has only five columns, while the rest have six.*

5.2.5 Image Lines and Poses Known, Find Model Lines

The poses and points are solved simultaneously without using lines. Lines could be included, as they are when solving for the pose alone, but the problem is complicated enough as it is. Once the poses are solved, however, each 3D line can be solved independently. Since the work of matrix multiplication is proportional to the cube of the size of the problem, many small problems are much faster than one large system of equations.

Since the image points are derived from vertices between image lines, one might expect that every model line could be solved by simply extending it between the corresponding model points. But not all image lines are used to produce model points at both ends, or even one end. Only enough points are needed to solve for the poses, they are not used during recognition at all. Because many endpoints are omitted, for uniformity, all lines are solved directly from the images.

This problem is very different from the pose-from-lines problem. The error is still the distance from the projected model line endpoints to the extended image line, but the variables are now the model line endpoints, not the pose. Also, each line is determined by itself, from multiple images. Because there are not multiple lines to impose offsets at a fixed distance in the image, the solution would quickly move the two model points very close together to avoid the orientation constraint in each image. To prevent this, the line endpoints in the images are needed. Since they are unstable, the tangential distance along the line between the projected endpoint and the measured endpoint is weighed less heavily in the equations (by a factor of five). The tangential error, like the perpendicular error, is just a linear combination of the x and y errors for a point. The partials are therefore the dot-product of the unit tangent vector with the partials of $\vec{x_{mi}}$ with respect to $\vec{x_m}$.

If both the tangential and perpendicular errors were included with equal weight, the line equations would just be applying a rotated version of the point constraints at each endpoint. Since the point constraints are rotationally symmetric, it would be exactly the same as two separate point equations, independent of the orientation of the line. However, the rotation of the line constraints is preserved so that the tangent error can be weighed less heavily. As with image points, the line must be visible in at least two images.

Chapter 6

Experimental Results

In order to determine the efficiency and accuracy of region-based grouping and interpretive matching, Reggie was run on twelve images with the mouse model. The image edges are shown in Figures 6.1 and 6.2. The extracted line features are shown in Figures 6.3 and 6.4. The twelve images were taken under natural office conditions (fluorescent lighting, typical objects and placements), and contain the mouse object under a variety of viewing directions, at various locations in the image, and over a range of scales. Also, the mouse is partly occluded by other objects in most of the images. Most of the images contain a single instance of the mouse, but three of them contain two mice, and one contains none. The task is to recognize all instances of the mouse model in each image. For each instance, the system should determine the pose of the model, find the correspondences between the model and image features, and calculate a score indicating the strength of the match.

I have made an effort to report all results honestly, so that both the strengths and weaknesses of this approach may be understood. This is done at the risk of having the success underestimated, when compared with reports of other systems that emphasize only their strengths. In particular, the twelve images presented here are the entire set of test images, not just the best ones. They are intended to include some cases that are too hard to recognize, in order to examine the failure modes. It is also important to remember that Reggie is a complete system, with many stages, and its performance as a whole is only as strong as its weakest module. However, only two of the modules were the focus of this research. The other modules take much simpler and more standard approaches to problems that are still not solved, by any means, and would benefit greatly from further research. Finally, the domain must not be underestimated. Finding all instances of a 3D model in a natural 2D scene is a visual task that very few systems have addressed with any success, to date [Goad 86] [Lowe 87] [Huttenlocher 88].

In judging the performance, two general properties are measured: accuracy and

efficiency. The system is accurate if the task is accomplished correctly. More specifically, if the pose is very close to correct, and a large percentage of the correspondences are correct, and most of the possible correspondences are found, and the score is sufficiently high, then the instance of the model has been recognized. It would be ideal if recognition were achieved whenever a human could identify the mouse in the image, but this is far beyond the current capabilities of vision systems in this domain. Instead, the evaluation of accuracy must be subjective: how "difficult" can the image instance be before the system fails to recognize it? One possible indication of the difficulty is the percentage of model lines that are visible in the image, so this quantity is counted (by human sight) for each instance. The results are discussed in Section 6.1.1.

In addition to overall accuracy, the performance of each module more specifically reflects the success of the region grouping and interpretive matching strategies. For grouping, the accuracy is indicated by the number of groups found on the model, and the amount of overlap between the image groups and the model groups. Some results have already been presented in Section 2.4, and some further results are presented in Section 6.2.1. In Section 6.2.2, the accuracy of interpretation is analyzed. Accuracy is indicated by the number of correctly interpreted group matches compared to the potential number of correct matches, given the image and model groups. The results are analyzed case by case, both for correct and for incorrect interpretations.

In Section 6.1.2, efficiency is evaluated. Since verification is a costly process, the number of match hypotheses that must be verified is an indication of the amount of work done. This is compared to the total number of possible combinations of minimal match hypotheses. In finding the matches, the efficiency of grouping is indicated by both the small number of groups produced, and by the small amount of time required to find them. The efficiency of interpretation is measured by the number of nodes explored.

In Section 6.2.4, the accuracy of the verification module is discussed. Although it was not one of the primary modules of this research, it is necessary to demonstrate that the earlier modules give adequate support for verification. In Section 6.3, practical improvements and conceptual extensions to the current work are discussed.

6.1 Overall Performance

6.1.1 Accuracy

The following table indicates that Reggie correctly found eight of the fourteen instances of the mouse model that are visible in the images. This result alone does not clearly indicate the accuracy of the system, since it does not describe the difficulty of

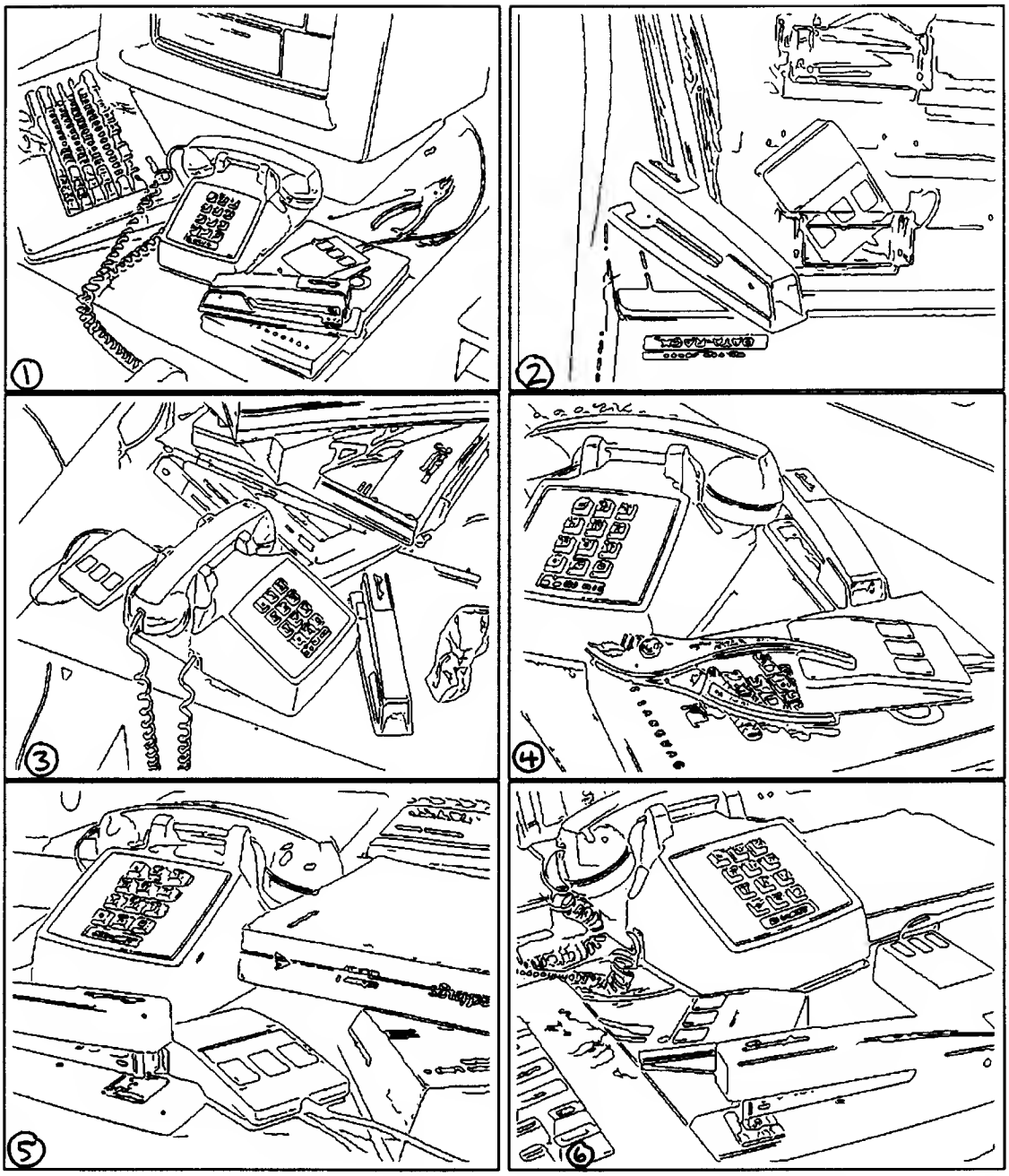

Figure 6.1: *The edges from Images 1 through 6 used in the experiments. Six more images appear in the next figure.*

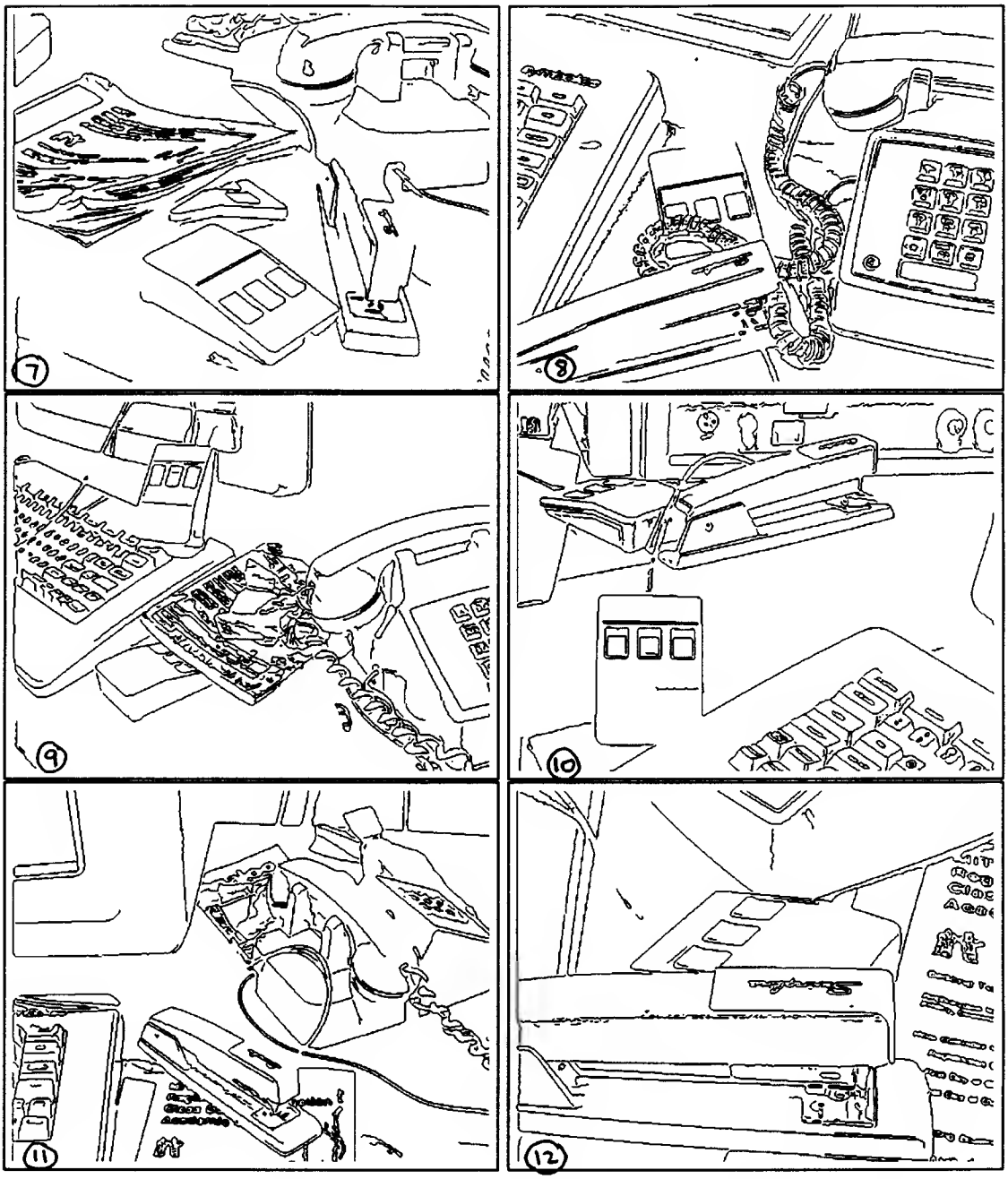

Figure 6.2: *The edges from Images 7 through 12 used in the experiments.*

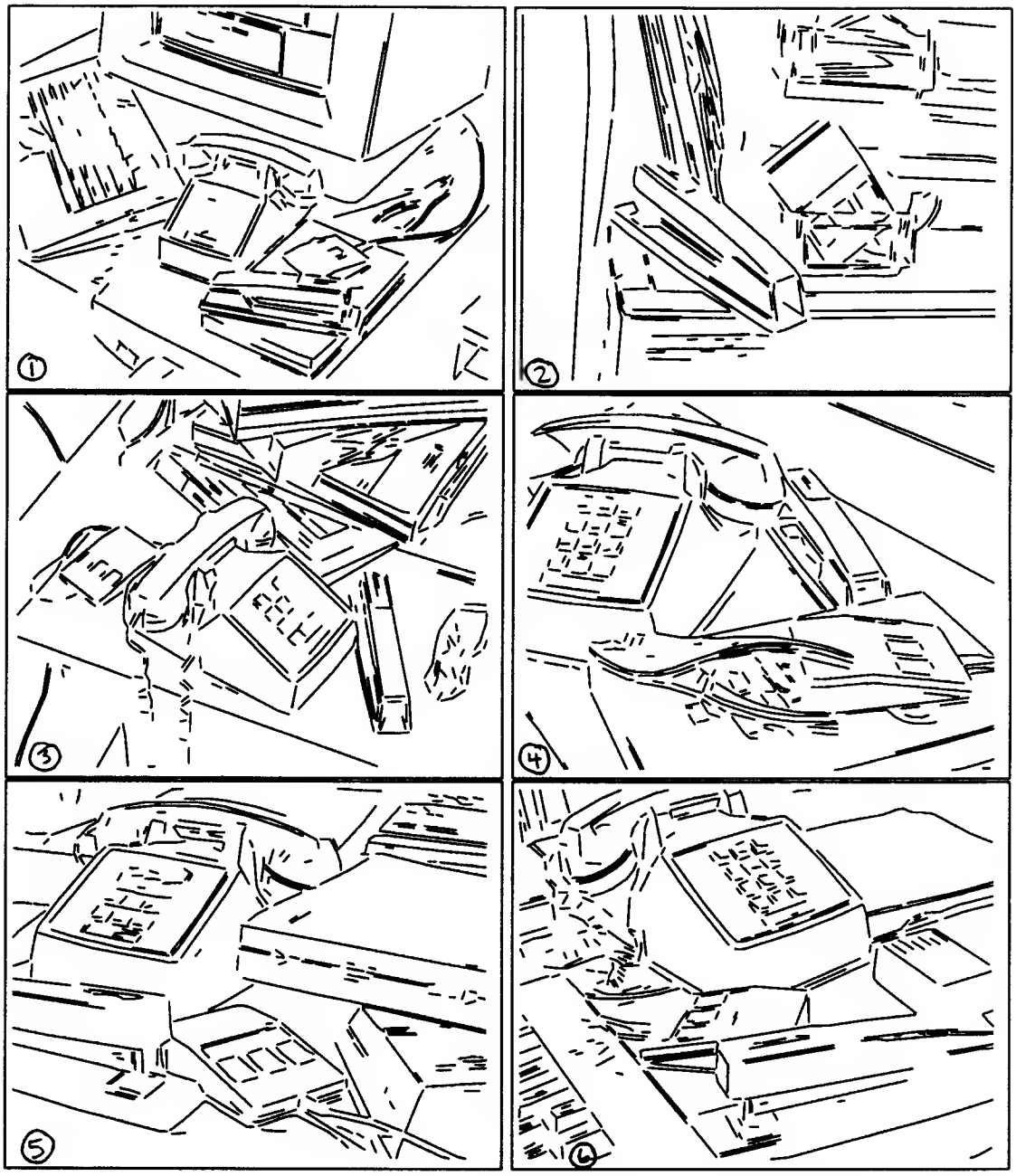

Figure 6.3: *The lines from Images 1 through 6 used in the experiments. Six more images appear in the next figure.*

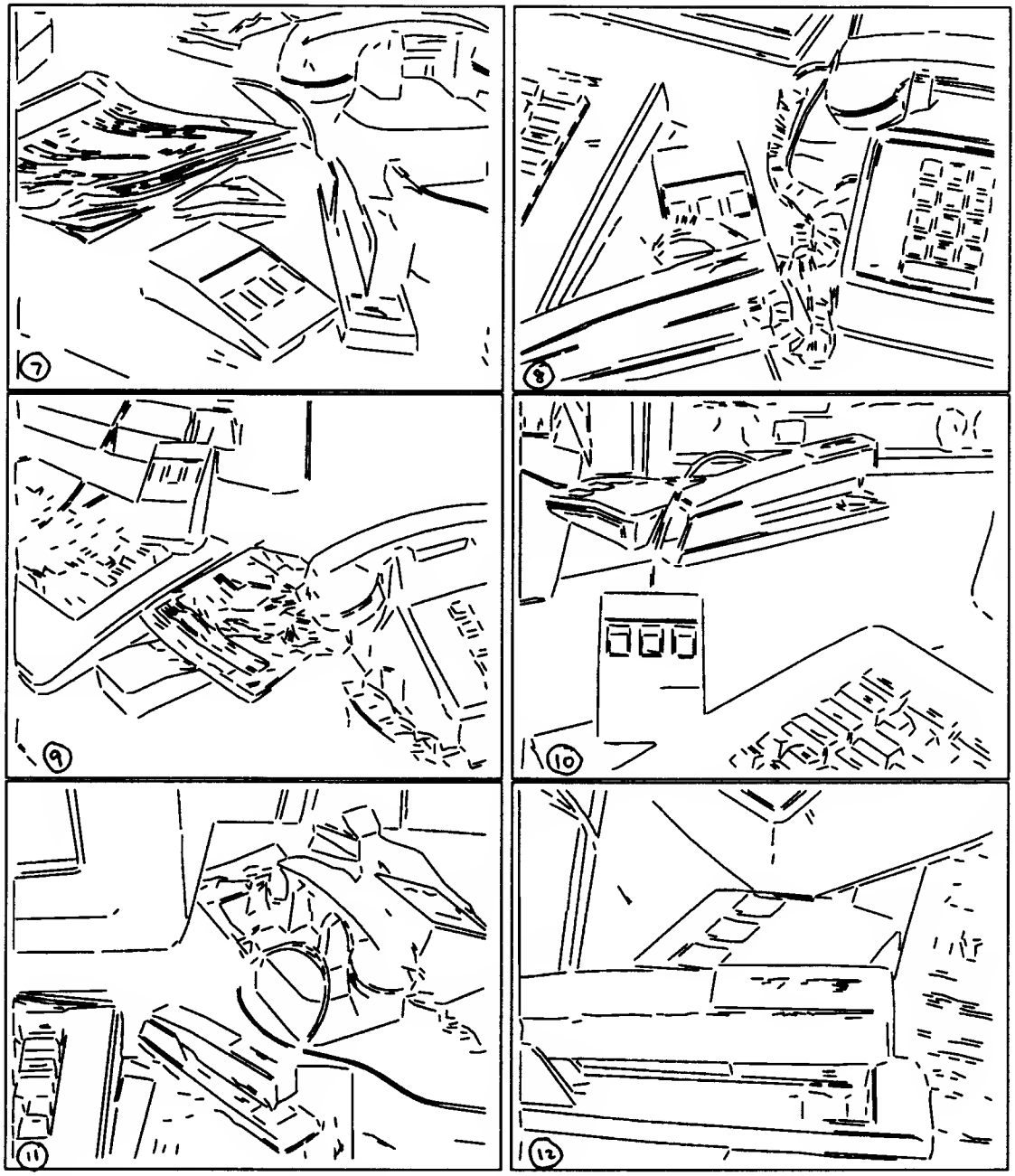

Figure 6.4: *The lines from Images 7 through 12 used in the experiments.*

the instances. Ideally, one would like to know the conditions under which recognition can be achieved, and the situations that cause failure. A crude indication of the difficulty of an instance is the number of model lines that are wholly visible in the image, as listed in the table. Another indication is the number of region groups found on the image of the model, which is also listed. The correct matches, once verified, are unambiguously close to the correct pose and feature correspondence. The verification scores are given in Section 6.2.4.

Recognition Accuracy			
Image	N model lines wholly visible (out of 33)	N groups in image of model, size ≥ 3	Verified discovery?
1	8	4	yes
2	12	4	no
3	14	5	no
4	17	5	yes
5	18	9	yes
6	12	6	yes
7a	20	8	yes
7b	9	5	no
8	11	5	yes
9a	15	4	yes
9b	9	4	no
10a	17	11	yes
10b	16	5	no
12	14	6	no

To better understand the conditions for success, each instance is briefly described below. Refer to figures 6.3 and 6.4 to see the line features available for recognition, and Figure 6.5 to find the mouse groups referenced by number.

In Image 1, the mouse is very small and most of the lines have been omitted. This would normally be a case in which Reggie would not succeed because so many groups are ruptured by the line omissions. However, by chance, the bottom of the middle button takes the place of the bottom of the button frame, and model group 109 leads to verification.

In Image 2, the mouse is half "occluded" by a transparent bin. Region 111 is mostly visible, but a small extra feature near the edge of the bin prevents the proper formation of the region, for unknown reasons. Region 118 might have been recovered if the occlusion were detectable, but the transparent edge disappears as it crossed the region boundary.

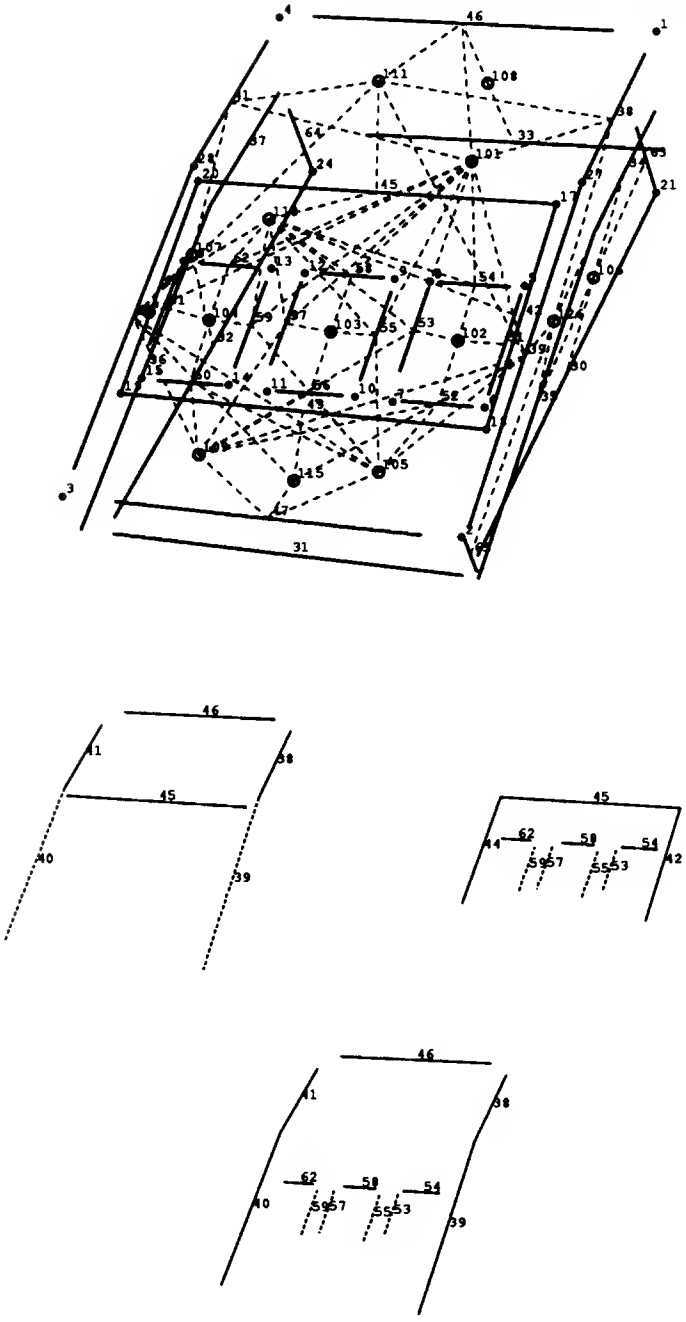

Figure 6.5: *The model region groups, lines, and points, by number. Also shown in isolation are model groups 111, 118, and 101, which account for the instability of line 45.*

In Image 3, again, the model is small and many lines are missing. The omission of the top of the left button prevents region 101 from matching properly, and the omission of the bottoms of all three buttons ruins regions 105 and 109.

In Image 4, both complete buttons lead to correct discovery despite broken lines, while the omission of the bottom of the right button prevents region 105. The pliers split region 111 without detectable occlusion, because the tip of the handle does not block the continuation of any line in the group.

In Image 5, all six of the visible groups lead to recognition, including an occluded group (111) and several model lines that are broken into two image lines. Region 111 also includes an extraneous feature which is correctly interpreted due to its length. Region 109 is missing a member, but this fact is successfully anticipated in the model.

In Image 6, only button region 104 is intact, with a broken line that is correctly detected. Both other buttons have a stray line added from the small wedge at the bottom of each button, which is not included in the model. Regions 111 and 118 are opened up since the top of the button frame is only partly detected.

In Image 7, the mouse pose is almost identical to Image 5, which is a very successful image. However, due to slight differences in line length, all three buttons are eliminated because the lines do not coterminate at the tops. Luckily, both regions 111 and 109 are intact, with some omissions anticipated. Region 118 is only missing one part of the button frame. The second (upper) instance is barely visible, but region 127 would match if not for an extraneous feature from the mouse that is omitted from the model.

In Image 8, regions 118 and 102 are visible and are correctly matched, despite four omissions and a broken line. Region 111 is connected to the background because the edge is not detected. Button region 103 is occluded, but the occluder does not produce connected lines.

In Image 9, central line omissions on the buttons leave only one group intact, region 115, which successfully leads to recognition despite a broken line. The second (lower) instance has missing button lines and button frame, preventing region 111 from matching the visible portion.

In Image 10, the button groups are ruined by many extra internal lines, but the presence of the bottom (top in image) of the button frame allows region 115 to be found after interpreting two broken lines. The partial presence of bottom of the frame and the occlusion that causes no edge prevent region 101 from forming properly. The second (upper) instance has four groups with sufficient similarity to the model, despite its unusual pose. However, partial lines and extra lines ruin all four potential matches.

In Image 11, there is no mouse. This image is included to test the amount of time required to determine that there are no instances of the model.

In Image 12, out of five sufficient groups, none can be matched. Buttons 104 and 103 have extra features, as in Image 6. In button region 102, part of the occlusion

is detected, but the other part is not close enough to be connected. Region 115 is not properly merged, for unknown reasons. Region 101 suffers from an occlusion that does not cause an edge in the image.

In summary, the region grouping module is fairly successful at finding sufficiently stable groups, and broken lines and certain feature omissions are often successfully interpreted. However, even a single extra or omitted line feature prevents recognition in many cases. The redundancy of the groups allows the entire system to succeed despite this sensitivity. Occlusion detection is most commonly foiled by the lack of edges and line connectivity. If line detection were more stable, and if some of the extraneous model features could be added to the model, accuracy could be improved considerably. Several specific improvements are presented in Section 6.3. The accuracy of specific modules is presented in Section 6.2.

6.1.2 Efficiency

Grouping and interpretive matching greatly reduce the number of matches that needed verification, in comparison to all possible combinations of three matches. This reduction is the primary achievement of the Reggie system, and both modules play a large role. In the table below, the number of line features (of length at least 8 pixels) is given for each image. The number of triples that could be formed from these lines is listed next. Three is the minimum group size to perform alignment, although it may not be large enough to provide sufficient accuracy, depending on the image error. By comparison, the number of region groups is smaller by a factor of at least one hundred thousand, and the average size of a group is significantly larger, providing more information to determine an accurate pose for verification. The minimum size of a group is set at four lines. (Grouping is also performed in the model, as shown in the second following table.) Given the specific model and image region groups, the number of combinations of matches within them is calculated. The number is calculated assuming that the circular order constraint would be enforced, but that all combinations of feature omissions would be tried within each model/image group pair. In contrast, the number of interpreted matches shows how few match sets are formed when occlusion and unstable features are explicitly detected and pose-invariant constraints are applied. The number of nodes explored is the number of hypothetical matches or feature omissions made during the interpretation. It is an indication of the complexity of performing interpretation. Since each node takes approximately two milliseconds to execute, interpretation takes roughly two to ten minutes. But, since each match takes approximately one second to verify, the small number of matches produced by interpretation has an equally important effect on the overall runtime.

Image	Num. image lines	All groups of three in image	Num. region groups	Ave. group size	All matches to model within groups	Num. interp. matches	Number of nodes explored
1	401	11 million	65	6.6	850 million	287	46000
2	293	15 million	60	6.8	180 million	65	67000
3	459	16 million	90	7.1	610 million	806	58000
4	431	13 million	62	7.4	87 million	128	54000
5	398	10 million	75	8.1	825 million	2269	720000
6	470	17 million	76	7.3	150 million	299	100000
7	454	15 million	69	8.2	2500 million	356	110000
8	521	23 million	68	7.6	580 million	278	160000
9	477	18 million	70	7.1	990 million	232	140000
10	374	9 million	62	7.5	1300 million	427	71000
11	432	13 million	72	7.1	306 million	440	130000
12	246	2 million	67	8.3	103000 million	363	360000

Model	Number of lines in model	All groups of three in model	Number of region groups	Average group size
Mouse	32	4960	9	6.6

These results indicate that both grouping and interpretation are able to provide large reductions in the number of matches. They also indicate the drawbacks of exponential sensitivity to group size. In image 12, the hundred-fold increase in the number of possible matches within groups is not an error. In that image, there is one group with thirty members due to a weak edge that is broken in many places. The number of matches depends exponentially on the size of the groups, and the image demonstrates the dangers of combinatoric explosion. With interpretation, we can see that the number of nodes explored and the number of matches produced have a fairly wide ranges of values. This is also due to exponential sensitivity: even though the interpretive matching tree is constrained, it still has a number of nodes that is exponential in the size of the group. In the race between splitting off new branches and constraining them, it can be a very close finish. For model groups that do not have parallel, perpendicular, or coterminating lines, fewer constraints can be applied and the number of explored nodes can rise suddenly. This is especially true when the interpretation involves many occlusions, colinearities, and marked skips.

The exponential sensitivity became evident during testing. Originally, the pseudo-perpendicularity constraint (described in Chapter 3) was not included. When an image group happened to have a large number of parallel lines, and it was matched with a model group with no cotermination constraints, the search exploded into a very

large number of nodes just from normal occlusion and feature instability branches. The addition of the pseudo-perpendicularity constraint brought the search well back within reasonable bounds. Together with parallelism, the constraints now seem to span the space of image configurations fairly effectively.

The results also show that efficiency is not reduced by multiple instances of the mouse in an image. In images 7, 9 and 10 there are two instances of the mouse, and in image 11 there are no instances. The number of interpreted matches remains typical in these images, as does the accuracy.

6.1.3 The Mouse Model

Since only one model was tested, it is worth discussing the possible influence of its particular features on the results. The mouse was chosen because it is within the domain of this approach: it has smooth surfaces without visual texture (patterns or text), and is fairly polyhedral. In addition, it has some specific properties that are helpful and some that are problematic. Like many man-made objects it has a large number of parallel and perpendicular lines. These provide more constraint during search. Also, the buttons provide three strong small groups that are particularly recognizable. However, most of the 3D edges, where faces meet, are rounded. This causes variation in edge location and weaker intensity gradients in the image. It also prevents lines from extending fully to vertices. Therefore, the cotermination constraint is not available in many cases. Further, the mouse has several physical properties that cause unstable features. The square frame around the buttons is only a slight recess, and often does not cause edges. The buttons are not flat—they have a raised wedge at one end which is too small to include in the model, but which often breaks model lines or replaces them with near misses. Together, I believe that the particular characteristics of the mouse balance to provide a fairly typical model within the domain.

6.2 Accuracy of Individual Modules

6.2.1 Grouping

The accuracy of grouping as an individual module is indicated by the number of groups with desirable properties. In Section 2.4, the number of groups from single objects is counted for a single image. In this section, various properties of groups that come specifically from instances of the modeled object are reported. In all cases, the size of a group is measured by the number of corresponding model features, not the number of image features. For example, when a model line is broken into two

lines in the image, the image group that contains the image lines is credited with only one line, not two. In the following table, four kinds of groups are counted in each image, by human sight. First, image groups of size three or more are counted to get a feeling for the relative potential of a recognition scheme that would use groups of minimum size. Reggie uses groups of size four or larger, which are also counted. Since groups are also formed in the model, the number of image groups with an intersection of at least size four with any model group is counted. These are the groups that would have led to successful recognition if interpretation had not been used, but if all combinations of matches within groups were tried instead. Finally, the number of groups that actually lead to correct matches is shown, but this reflects primarily on the interpretation module, not the grouping process.

Grouping Success				
Image	N groups due to model, size ≥ 3	N groups due to model, size ≥ 4	N groups intersect, size ≥ 4	N groups correctly matched
1	4	2	2	1
2	4	2	1	0
3	5	2	2	0
4	5	5	4	2
5	9	7	6	6
6	6	4	3	1
7a	8	8	6	2
7b	5	2	2	0
8	5	3	3	2
9a	4	2	1	1
9b	4	2	1	0
10a	11	4	2	1
10b	5	4	4	0
12	6	5	5	0

In all cases, at least one image group is present with sufficient similarity to a model group, and in many cases more than one is found. This indicates that the grouping module is quite adequate for this domain. Because not all groups are successfully interpreted, the presence of multiple groups on the object is often very important for finding the model instance.

6.2.2 Successful Interpretations

The test images include a wide variety of feature instabilities and occlusion situations, and Reggie is often able to overcome them. In the following table four categories of

image feature difficulties are defined and the number of cases of successful interpretation is listed for each category.

Modes of Successful Interpretation	
Image difficulty	Number of occurrences
Model feature broken but colinear in image	17
Model feature omitted from group in image	10
Extraneous image feature	1
Occlusion	1

There are sixteen correctly matched groups in all, but several overcome multiple difficulties, while others are matched directly without the need for interpretation. The simplest interpretation is the most needed and the most successful: when colinear pairs of lines appear in the image, in addition to a direct match, the longer one is matched and the shorter one is skipped. This corrects for many occasions where the model line is split in two parts in the image.

The second most commonly corrected difficulty is missing model features. There are three reasons why a feature might be missing: it might be absent from the edges entirely (due to insufficient intensity contrast), or it might be too short to be considered a stable feature, or it might not be included in a group. In all ten correctable occurrences, unstable grouping is responsible for the missing feature. This is corrected only through anticipation, not from evidence in the image. Features that are likely to be missed in the grouping process are marked in the model group. During matching, these model features are both matched and skipped. A forced split in the match tree is somewhat expensive, but it provides valuable accuracy in this case. Sometimes grouping omissions are correlated. For example, from one viewing direction certain lines may be more likely to be omitted, while from other directions a different set tend to be left out. In this situation, an alternative to marking all skip-features is to form multiple model groups, containing many of the same features but with different ones marked. Overlapping groups are also used to address radical grouping changes due to the omission of a feature.

It is possible for a marked model feature to account for the other two types of image feature omission, but this never occurs in any of the test images. The situations in which model lines do not cause any edges are much harder to anticipate or detect from the surrounding edges. However, features that are detected in the edges but are too short are often erroneously omitted from the search. This problem is significant and could be addressed, as discussed in Section 6.3.

Extra features in the image are also very difficult to interpret from the image alone. Currently, the only indication that a feature is unstable is its length. Features

in the length range from 8 to 10 pixels are marked unstable, and both matched and skipped during interpretive matching. In the images, this test correctly omits one extraneous image feature, but misses three others. The instability criterion is too arbitrary and not well adjusted to the actual causes of extraneous features. It is likely that more accurate interpretation could be accomplished by inspecting the grey levels directly, or the gradient magnitude, as discussed in Section 6.3. Another possibility is to include all dubious features in the model, but mark them as unstable. This is a direct tradeoff between efficiency and accuracy.

The lack of contribution from occlusion interpretation is unexpected. Locally, the detection of an occlusion condition is fairly accurate. The lack of successful occurrences is partly due to the lack of occurrences at all. In almost all the images the mouse is occluded to some degree, but there are only six groups in which occlusion is present. One of these is correctly interpreted. The other five are either undetectable (due to the absence of edges) or too complicated for the simple interpretation scheme. Apparently, it is rare that an occluded group retains enough of the original group and has detectable occlusion. Usually the occlusion breaks the group or joins it seamlessly at one end. However, more sophisticated occlusion analysis is probably worthwhile, especially because of its potential application in verification, as was discussed in Section 4.2.3.

6.2.3 Analysis of Failures in Interpretive Matching

Because of the redundant groups on the object, failures in matching do not always lead to failures in recognition. However, understanding the modes of failure is crucial to improving performance. The success of each group seemed independent from the success of its neighbors, so improving the likelihood of success in every case would certainly increase the accuracy of the system as a whole. Below, some failure modes are identified and listed with their frequency in the experimental images. As with the modes of success, some group matches encountered multiple difficulties, and all are counted. An occurrence is counted if an image group contains lines from at least four grouped model lines, but is prevented from successfully matching due to the difficulty.

Modes of Failure	
Image difficulty	Number of occurrences
Grouping: feature left out of group	3
Grouping: missing feature, bad merge	2
Grouping: extra feature, bad split	2
Occlusion	5
Model feature missing in image	4
Extraneous image feature	3
Missing line relation	4
Non-colinear split feature	1
Model features merged in image	1
Other	1

Despite the general success of grouping, it is responsible for seven of the failures. In three cases, it neglects to include a line that is not anticipated in the model as a likely omission. In two cases, the absence of a feature in the image causes the group to merge in an unexpected way. This includes the case of a line that is only half detected in the image, so it becomes a member of the group but also allows the group to spread beyond it. In two cases, the presence of an unmodeled line splits a group into two smaller groups. In all of these cases, if all possible matches to each model group were tried the correct match would be found. It is the need to avoid those combinatorics that forces much fewer matches to be tried. When attempting to match features one-to-one within the groups, it only takes one unanticipated extra feature to prevent the correct match from being discovered.

Occlusion accounts for five failures. Because all detected occlusion "starts" and "ends" are both heeded and ignored during matching, even the most complicated occluder should be detected if the beginning and end of its extent are detected. Despite a fairly good percentage of local occlusion detection, there are enough errors to cause these five failures. In three cases, this is due to a gap between lines from the occluding object. Occluding lines are not detected unless they are connected by cotermination or colinearity. In two cases, the boundary between the foreground and background objects does not have a sufficient intensity gradient to cause an edge. At the occlusion points, the edge continues smoothly from one object to the other. To detect this kind of occlusion would require a more global interpretation.

Missing and extra image features cause seven failures in the test images. Unless anticipated, they prevent the discovery of the correct match, similar to the grouping omissions and additions. Unlike grouping instabilities, they are due to feature extraction errors and edge detection errors, or simply due to the fact that illumination and reflectance provide no indication in the image of the 3D edge. A common cause

of feature omissions is the line length threshold. When a visible model line is broken into two short image lines at a small scale, it is common for the lines to be too short to meet the stability criterion. The issue of stable lines is considered further in Section 6.3.

Four failures are caused by missing line relations, cotermination in particular. The model relations between lines are required to be present in the image in order to allow a match. This is the primary constraint on the search. In these four cases, the image line endpoints are not close enough to each other to pass the cotermination test, although their corresponding model lines come to a vertex. In three of these cases, the mouse is fairly large in the image. If the model were a perfect polyhedron and the image edges formed perfect polygons, then the scale would not matter. In realistic conditions, corners are where the edge detector tends to wander from the path, and the small details visible at a larger scale also cause edges to deviate. The edges curve enough near the vertices to prevent the lines from extending fully.

There are eighteen cases in which a model line appears as two image lines, but only one in which the resulting two image lines fail the colinearity test. The failure is simply due to a weak edge that is very irregular. However, there is also one case in which two model lines appear as a single line in the image, and this situation is not considered by the current interpretation scheme. While any two lines may appear colinear in the image, it should be possible to mark model lines that are more likely to be colinear. This is different than marking them with a colinear line relation, since that would require the image lines to be colinear. Instead, the lines must be allowed the possibility of colinearity, just as unstable model group members are allowed the possibility of omission in the image. Similarly, only the most likely occurrences can be anticipated in this manner.

In the category of "other", the constraint application module is confused by two line segments which coterminate at both ends. This could be fixed by including more information when the line relations are detected.

From this detailed inspection of the failure modes it seems that the general concept of interpretation is sound, but there is room for improvement. There are some image obfuscations that might be very difficult to interpret. Anticipating these situations by marking the model with likely trouble spots has limited utility. Marking the model extensively is not in the spirit of interpretation—it approaches the method of forming all combinations. But there are many situations that may benefit by more extensive interpretation. In Section 6.3, I offer some suggestions for future work in this direction.

6.2.4 Verification Scores

Although verification is not the focus of this work, the ability to verify a hypothesis is one of the pressures on hypothesis formation. For example, the number of features in a group affects the accuracy of the pose. Therefore, the verification scores are included below. The highest incorrect score is also included to estimate the discrimination of the scoring function. When the correct match does not have the highest score, all higher scores are included.

Verification Scores			
Image	N groups correctly matched	Scores of correct matches	Highest scores of incorrect matches
1	1	.38	.42 .38
2	0	not found	.29
3	0	not found	.37
4	2	.53 .47	.44
5	6	.56 .54 .53 .53 .52 .51	.38
6	1	.44	.46 .45 .40
7a	2	.61 .56	.44
8	2	.54 .51	.50
9a	1	.46	.44
10a	1	.47	.51 .49 .45
11	-	no instances	.51
12	0	not found	.30

Despite some weaknesses of the verification scoring function, the correct matches score well enough to prove that they support verification by alignment. The correct matches are often the best score, and are always included in the highest five scores. The scoring function is definitely not precise enough to set a threshold for discovery, however. Working backwards from the results, a threshold at 0.5 would produce four correct detections, three incorrect detections, and miss four correct matches. A threshold of .45 would produce six correct detections, seven incorrect detections, and miss two correct matches. This is not quite an acceptable level of discrimination, and it does not allow for heavier occlusion. Occlusion lowers the scores of correct matches without affecting the scores of incorrect matches.

Overall, these results indicate that match-independent information has great potential for improving efficiency while achieving reasonable accuracy. However, the images also push the modules to their limits, revealing areas that could be improved, as discussed in the next section.

6.3 Possibilities for Future Work

Within the framework of grouping and interpretation there are many potential improvements to the current Reggie system. Some have already been mentioned above and in Section 4.2.3 on verification.

6.3.1 Scale Invariance

If there is one general effect that caused more problems than any other, it is scale. In the test images, the model appeared between eighty and three hundred pixels in width. Even over a reasonable range of scales, lines appear and disappear, are broken and joined, and change their separation. The tight constraints upon which Reggie relies do not leave room for much variation in features or their relations. The fixed line-length limit caused many feature omissions, the fixed limit on tracing during grouping caused several grouping omissions, and the fixed cotermination distance caused several line relation omissions. If these cutoffs could be made more adaptive to scale, performance would improve. To remove the line length threshold, two options are available. First, the threshold could simply be removed, and all short lines could be marked as unstable. They would then be both matched and skipped during interpretation, at a significant cost in efficiency. But this avoids the problem: why are these lines unstable? A closer look at the underlying edges should prove fruitful, as described next.

6.3.2 Smooth-Curve-Section Features

Lines that come from smoothly curving edges should never be matched to model lines, no matter how long the fitted line segments are. In fact, in addition to line segments, smooth curve sections could be an important additional feature type, and this is the second way to avoid a line length threshold. The concepts of grouping and interpretation could easily include smooth curve sections; those concepts are very general. The problems with curve features are more practical. The endpoints are unstable, and the shape varies with viewpoint, while lines vary only in length. Curves are therefore hard to match reliably. The interpretive matching module would have to anticipate several kinds of distortions, but I believe it would be possible. An important advantage to their inclusion is that edges would be represented continuously. Currently, gaps between lines due to curved edges are a prominent difficulty for occlusion interpretation and line relations. With curves, connectivity would be preserved. This is such an important grouping trait that it might be worth detecting curve sections simply for interpretation, without including them in the matching process.

A more sophisticated use of curve section features would be to model and recognize curved surfaces. This would be an important extension to the domain. The process of recognizing objects from their occluding contour has been explored as a field of its own ([Chien and Aggarwal 87], for example), and recently with a new approach by [Basri and Ullman 89].

6.3.3 Occlusion

Despite the relatively unsuccessful performance of occlusion analysis during tests, I believe it is important and could be improved. As mentioned above, the addition of curve section features to capture continuity is one key improvement. Continuity could be added to colinearity and cotermination as an indicator that the occluding features continue outside of the group. This could increase the number of correct detections significantly.

A more intricate analysis of occlusion would also be beneficial. Currently there is no means for handling complicated situations, and only limited use of geometric relations. With more careful analysis, the number of false occlusion detections could be reduced. False occlusion detections not only cause a branch in the search, they also prevent the application of constraints at one end of the line, and are therefore particularly costly. Based on an inspection of false occlusion detections, several geometric relations seemed worth investigating. First, only the closest lines should be considered as possible occluders. Often, a nearby line blocks the possibility of further lines being involved, but this fact is ignored. Second, a line that crosses between two line endpoints should prevent the formation of a cotermination relation. Third, if a line is colinear with a neighbor, the neighbor should not be counted as an occlusion of the line, no matter what other lines are present.

To increase the number of correct detections, gaps between consecutive lines should be explored. The interpretation of occlusion heavily depends on the continuity of the edges around the group. When there is a gap, the current implementation does nothing. A gap can be an indication of occlusion, feature instabilities, or grouping instabilities. Its interpretation involves an analysis of the surrounding features, which has not yet been developed. Another possibility is to try to fill the gaps earlier in the visual process, as [Ullman and Sha'ashua 88] have done, for example.

6.3.4 Unstable Features

As discussed above, many lines are lost due to the line length threshold. Also, some are missing because the edges are not detected. It is possible that their absence may be interpreted from the surrounding features. Another alternative is to inspect the image grey-levels directly, both to find missing lines and to detect extraneous lines.

In the edge detector a threshold is set on the gradient magnitude. This threshold could be reduced and the weaker lines could be identified as unstable. Of course, this increases the number of branches during search. Grey-level analysis may also be used to identify different types of edges, such as those due to shadows or surface markings, as several researchers have investigated ([Herskovitz and Binford 70] or [Witkin 82], for example). This kind of analysis could be used to detect likely extraneous features.

6.3.5 Additional Constraints

If a set of model features are coplanar then there are relations beyond cotermination, parallelism, and pseudo-perpendicularity that are conserved under projection. Since many of the model groups are coplanar on a semi-polyhedral model, this might be a valuable additional constraint. However, the conserved values can only be extracted from four or more points, so the endpoints of lines would have to be used in order to apply the constraints to pairs of lines. Alternatively, the constraints could be applied to triples or quadruples of lines, but the combinatorics of doing so may be prohibitive.

6.4 Summary

The efficiency of region grouping and interpretive matching is very encouraging, as indicated by the very small number of matches that require verification. The execution times of the modules themselves are also very good, and seem fairly insensitive to the number of lines or groups in the image. The size of the largest group can have some influence on execution time, but the effect is limited by the model constraints. In addition, there are several improvements that might increase efficiency even further, at little or no cost to accuracy. These include more sophisticated line instability detection and more certain occlusion detection.

The accuracy of the system is sufficient to demonstrate the potential of the approach, but could be further improved. One of the weakest links is the line detector, which would benefit greatly by the removal of the length threshold and the addition of smooth-curve-section features. Region grouping is remarkably accurate, both in its ability to find groups from single objects, and to do so repeatably, so that the model groups match the image groups fairly closely. Many problems with the image features are successfully interpreted, especially broken lines and lines omitted from groups. But occlusion interpretation does not work as well as hoped. It seems to be on the brink of success, however, and is most affected by the line instabilities and disconnections that might be alleviated in future implementations. Finally, some extra and missing image features are not detected or anticipated, but, by considering more instabilities in the image and model, the accuracy could be increased at the cost of

efficiency.

Chapter 7

Conclusions

The Reggie system was constructed to demonstrate the viability of region-based grouping and interpretive matching. Both of these modules take advantage of match-independent knowledge about the world, the image formation process, and the modeled objects as a whole. This extra knowledge is part of the constraint that allows us to perceive a three dimensional world from a two-dimensional image of it. Almost all previous recognition systems have used the model shape as the primary constraint. Some have also used some match-independent knowledge to help reduce the search, but without emphasizing the concept or tapping its power. Lowe was the first to explicitly bring the concept of perceptual organization to model-based recognition, demonstrating that parallel lines could be used to form useful groups. In Reggie, the concepts are taken further: a new grouping strategy is introduced, and feature instabilities and occlusion are directly interpreted. These modules drastically reduce the search space because they alleviate the need to try all combinations of matches in order to account for other objects in the image, feature instabilities, and occlusion. Instead, these modules attack the matching problems head-on. When combined with pose-independent constraints during matching, only hundreds of match hypotheses remain for verification, instead of the billions that would have to be verified if all combinations of minimal matches were considered. The reduction is so deep that all satisfactory hypotheses may be verified without the need for ordering them or terminating the search prematurely. Thus, Reggie can find multiple instances of the model in an image with equal efficiency and accuracy. Despite the elimination of so many hypotheses, enough correct matches are preserved to recognize the modeled object in most of the tested cases.

Region-based grouping has proved to be very successful at collecting image features that are likely to come from a single object. A high percentage of the groups are from single world objects, and almost all prominent objects have at least one group entirely from them. The image groups are sufficiently accurate that initial model

match hypotheses may be limited to one group at a time, eliminating all matches across image groups. To increase efficiency, groups are also formed in the model, and the search for hypotheses is restricted to matches between one model group and one image group at a time. In this case, grouping must also be stable: there must be an intersection between a model group and corresponding features in an image group, large enough to verify by alignment. In experiments, the region groups have had very good stability and have always produced at least one image group with sufficient similarity to a model group. Furthermore, in addition to the reductions due to partitioning matches into groups, region groups preserve the circular order of features within groups.

Region groups are also particularly appropriate for the later stages of recognition. For verification, they are large enough to support an accurate pose estimate. For interpretation, region groups define an "inside"; an indication of where the object is. They also provide a connected boundary which can be inspected for intrusion from outside, due to occlusion.

The interpretive matching module demonstrates that occlusion and unstable features need not be accounted for by matching all combinations of features. By inspecting the image evidence and by understanding the model and its likely appearance, specific feature mismatches can be detected and anticipated. This allows search branches to be generated only when needed. When combined with pose-independent constraints from the model, interpretation prunes the search to a very manageable size before having to solve for a pose.

While Reggie clearly demonstrates the potential power of applying match-independent knowledge, experiments also point to areas that could be improved. There are several ways that efficiency could be increased, but improvements in accuracy are in greater need. Primarily, accuracy could be increased if the edge contours were utilized more completely. Currently, all edges are used to bound region groups, but for interpretation, matching, and verification only straight line segments are used. Curve sections should be included at least during interpretation, where connectivity plays such an important role. Second, the current implementation does not interpret gaps between features around a region. Gaps can indicate several relevant image conditions, but they require more elaborate analysis of surrounding features. These improvements are relatively minor but could have a significant effect on accuracy.

In a broader perspective, the general process of interpreting the scene deserves much more development in vision. The matching problem is far too vast to rely only on match-dependent model shape constraints, especially when large libraries of common objects are to be found. The match-independent information that allows us to quickly understand a scene is vital to reducing the search, and the Reggie system demonstrates that scene interpretation can be blended with feature matching to great advantage.

Bibliography

[Asada and Brady 86] Asada, H. and Brady, M., 1986. "The Curvature Primal Sketch." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):2-14.

[Ayache and Faugeras 86] Ayache, N. and Faugeras, O., 1986. "HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):44-54.

[Ballard 81] Ballard, D., 1981. "Generalizing the Hough Transform to Detect Arbitrary Shapes." *Pattern Recognition*, 13:111.

[Barrow and Tenenbaum 81] Barrow, H. G., and Tenenbaum, J. M., 1981. "Interpreting Line Drawings as Three-Dimensional Surfaces." *Artificial Intelligence*, 17:75-116.

[Baird 84] Baird, H., S., 1985. *Model-Based Image Matching Using Location.* MIT Press, Cambridge, MA.

[Basri and Ullman 89] Basri, R., and Ullman, S., 1989. "Recognition by Linear Combinations of Models." The Weizmann Institute of Science, Department of Applied Mathematics and Computer Science, Memo CS89-11.

[Besl and Jain 85] Besl, P. J., and Jain, R. C., 1985. "Three Dimensional Object Recognition." *ACM Computing Surveys*, 17(1):75-154.

[Biederman 85] Biederman, I., 1985. "Human Image Understanding: Recent Research and a Theory" *Computer Vision, Graphics, and Image Processing*, 32:29-73.

[Binford 81] Binford, T., 1981. "Inferring Surfaces from Images" *Artificial Intelligence*, 17:205-244.

[Blum 73] Blum, H., 1973. "Biological Shape and Visual Science, Part I." *Journal of Theoretical Biology*, 38:205-287.

[Bolles and Cain 82] Bolles, R. and Cain, R., 1982. "Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method." *The International Journal of Robotics Research*, 1(3):57-82.

[Bopp 78] Bopp, H., and Krauss, H., 1978. "An Orientation and Calibration Method fro Non-Topographic Applications." *Photogrammetric Engineering and Remote Sensing*, 44(9):1191-1196.

[Brooks 81] Brooks, R., 1981. "Symbolic Reasoning Among 3-D Models and 2-D Images." *Artificial Intelligence*, 17:285-348.

[Carlotto 87] Carlotto, M., 1987. "Histogram Analysis Using a Scale-Space Approach." *IEEE Transactions in Pattern Analysis and Machine Intelligence*, 9(1):121-129.

[Canny 86] Canny, J. F., 1986. "A Computational Approach to Edge Detection." *IEEE Transactions in Pattern Analysis and Machine Intelligence*, 8(6):34-43.

[Cass90] Cass, T. A., 1990. "Feature Matching for Object Localization in the Presence of Uncertainty." MIT Artificial Intelligence Laboratory Memo 1133

[Chakravarty 79] Chakravarty, 1979. "A Generalized Line and Junction Labeling Scheme with Applications to Scene Analysis." *IEEE Transactions in Pattern Analysis and Machine Intelligence*, 1(2):202-205.

[Chien and Aggarwal 87] Chien, C. H., and Aggarwal, J. K., 1987. "Shape Recognition from a Single Silhouette." *Proceedings of the (First) International Conference on Computer Vision (London)*, 481-490.

[Clemens 86] Clemens, D., 1986. The Recognition of Two-Dimensional Modeled Objects in Images, S.M. Thesis, MIT Department of Electrical Engineering and Computer Science.

[Clowes 71] Clowes, M. B., 1971. "On Seeing Things." *Artificial Intelligence*, 2(1):79-115.

[Connell 85] Connell, J. H., 1985. "Learning Shape Descriptions: Generating and Generalizing Models of Visual Objects." MIT Artificial Intelligence Laboratory Technical Report 853.

[Fischler and Bolles 81] Fischler, M. A., Bolles, R. C., 1981. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography." *Communications of the ACM*, 24(6):381-395.

[Forsythe, Mundy, Zisserman and Brown 90] Forsythe, D., Mundy, J., Zisserman, A., and Brown, C., 1990. "Invariance: A New Framework for Vision." *Proceedings of the Third International Conference on Computer Vision*, 598-605.

[Goad 86] Goad, C., 1986. "Fast 3D Model-Based Vision" in *From Pixels to Predicates: Recent Advances in Computational and Robotic Vision*, edited by A. P. Pentland. Ablex, Norwood NJ.

[Grimson and Lozano-Pérez 87] Grimson, W.E.L., Lozano-Pérez, T., 1987. "Localizing Overlapping Parts by Searching the Interpretation Tree." *IEEE Transactions in Pattern Analysis and Machine Intelligence*, 9(4):469-482.

[Hanson and Risemann 78] Hanson, A., Risemann, E., 1978. "Visions: A Computer System for Interpreting Scenes" in *Computer Vision Systems*, A. Hanson and E. Risemann, Eds. Academic, New York.

[Haralick and Shapiro 84] Haralick, R., Shapiro, L., 1984. SURVEY: Image Segmentation Techniques, *Computer Vision, Graphics, and Image Processing*, 29:100-132.

[Hochberg 57] Hochberg, J., 1957. "Effects of the Gestalt Revolution: The Cornell Symposium on Perception." *Psychological Review*, 64(2):73-84.

[Herskovitz and Binford 70] Herskovitz, A., and Binford, T., 1970. "On Boundary Detection." MIT Artificial Intelligence Laboratory Memo 183.

[Horn 71] Horn, B. K. P., 1971, revised 1973. "The Binford-Horn Line-Finder." MIT Artificial Intelligence Laboratory Memo 285.

[Horn 72] Horn, B. K. P., 1972. "VISMEM: A Bag of "Robotics" Formulae." MIT Artificial Intelligence Laboratory Working Paper 34.

[Horn 77] Horn, B. K. P., 1977. "Understanding Image Intensities." *Artificial Intelligence*, 21(11):201-231.

[Horn 87] Horn, B. K. P., 1987. "Relative Orientation." MIT Artificial Intelligence Laboratory Memo 994.

[Huttenlocher 88] Huttenlocher, D., 1988. "Three-Dimensional Recognition of Solid Objects from a Two-Dimensional Image." PhD Thesis, MIT Department of Electrical Engineering and Computer Science.

[Huttenlocher and Ullman 90] Huttenlocher, D. and Ullman, S., 1990. "Recognizing Solid Objects by Alignment with an Image." *International Journal of Computer Vision*, 5(2):195-212.

[Jacobs 88] Jacobs, D., 1988. "The Use of Grouping in Visual Object Recognition." MIT Artificial Intelligence Laboratory Technical Report 1023.

[Jacobs 89] Jacobs, D., 1989. "Grouping for Recognition." MIT Artificial Intelligence Laboratory Memo 1177.

[Kubovoy and Pomerantz 81] Kubovoym and Pomerantz, Eds., *Perceptual Organization* (Hillsdale, New Jersey: Lawrence Erlbaum, 1981).

[Lamdan, Schwartz and Wolfson 87] Lamdan, Y., Schwartz, J. T., and Wolfson, H. J., 1987. "On Recognition of 3-d Objects from 2-d Images." New York University, Courant Institute Robotics Report, No. 122.

[Lowe 85] Lowe, D. 1985. *Perceptual Organization and Visual Recognition.* Hingham, Massachusetts: Kluwer Academic Publishers.

[Lowe 87] Lowe, D., 1987. "Three-Dimensional Object Recognition from Single Two-Dimensional Images." *Artificial Intelligence Journal*, 31:355-395.

[Mahoney 87] Mahoney, J., 1987. "Image Chunking: Defining Spatial Building Blocks for Scene Analysis." MIT Artificial Intelligence Laboratory Technical Release 980.

[Marr 77] Marr, D., 1977. "Analysis of Occluding Contour." *Phil. Transactions of the Royal Society of London*, B(275):483-524.

[Marr and Nishihara 78] Marr, D., and Nishihara, K., 1978. "Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes." *Proceedings of the Royal Society of London*, B(204):301-328.

[Pavlidis 72] Pavlidis, T., 1972. "Segmentation of Pictures and Maps Through Functional Approximation." *Computer Vision, Graphics, and Image Processing*, 1:360-372.

[Roberts 65] Roberts, L. G., 1965. "Machine Perception of Three-Dimensional Solids" in *Optical and Electro-Optical Information Processing*, Tippett et al., Eds., MIT Press, Cambridge, Mass.

[Schwartz and Sharir 87] Schwartz, J. and Sharir, M., 1987. "Identification of Partially Obscured Objects in Two and Three Dimensions by Matching Noisy Characteristic Curves." *The International Journal of Robotics Research*, 6(2):29-44.

[Thompson and Mundy 87] Thompson, D. and Mundy, J. 1987. "Three-Dimensional Model Matching from an Unconstrained Viewpoint." *Proceedings of the IEEE Conference on Robotics and Automation*, IEEE Computer Society Press, 208-220.

[Turney, Mudge and Volz 85] Turney, J. L., Mudge, T. N., and Volz, R. A., 1985. "Recognizing partially occluded parts." *IEEE Transactions in Pattern Analysis and Machine Intelligence*, 7(4):410-421.

[Ullman 83] Ullman, S., 1983. "Visual Routines." MIT Artificial Intelligence Laboratory Memo 723.

[Ullman 86] Ullman, S., 1986. "An Approach to Object Recognition: Aligning Pictorial Descriptions." MIT Artificial Intelligence Laboratory Memo 931.

[Ullman and Sha'ashua 88] Ullman, S., Sha'ashua, A., 1988. "Structural Saliency: The Detection of Globally Salient Structures Using a Locally Connected Network." *Proceedings of the Second International Conference on Computer Vision*, IEEE Computer Society Press, 321-327.

[Wallace 87] Wallace, A., 1987. "Matching Segmented Scenes to Models Using Pairwise Relationships Between Features." *Image and Vision Computing*, 5(2):114-120.

[Waltz 75] Waltz, D., 1975. "Understanding Line Drawings of Scenes with Shadows" in *The Psychology of Computer Vision*, (P. H. Winston, Ed.), McGraw-Hill, New York.

[Wertheimer 23] Wertheimer, M., 1923. "Principles of Perceptual Organization" in *Readings in Perception*, David Beardslee and Michael Wertheimer, Eds., Princeton, NJ: 1958, 115-135.

[Witkin 82] Witkin, A., 1982. "Intensity-Based Edge Classification." *Proceedings AAAI (1982)*, 1:36-38.

[Witkin and Tenenbaum 83] Witkin, A., and Tenenbaum, J., 1983. "What is Perceptual Organization For?" *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*:1023-1027.

This blank page was inserted to preserve pagination.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1991	technical report

4. TITLE AND SUBTITLE

Region-Based Feature Interpretation for Recognizing 3-D Models in 2-D Images

5. FUNDING NUMBERS

N00014-86-K-0685

6. AUTHOR(S)

David T. Clemens

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Artificial Intelligence Laboratory
545 Technology Square
Cambridge, Massachusetts 02139

8. PERFORMING ORGANIZATION REPORT NUMBER

AI-TR 1307

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
Information Systems
Arlington, Virginia 22217

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AD-A259490

11. SUPPLEMENTARY NOTES

None

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution of this document is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Abstract. In model-based vision, features found in a two-dimensional image are matched to three-dimensional model features such that, from some view, the model features appear very much like the image features. The goal is to find the feature matches and rigid model transformations (or *poses*) that produce sufficiently good alignment. Because of variations in the image due to illumination, viewpoint, and neighboring objects, it is virtually impossible to judge individual feature matches independently. Their information must be combined in order to form a rich enough hypothesis to test. However, there are a huge number of possible ways to match sets of model features to sets of image features. All subsets of the image features must be formed, and matched to every possible subset of the model features. Then, within each subset match, all permutations of matches must be considered. Many strategies have been explored to reduce the search and more efficiently find a set of matches

(continued on back)

14. SUBJECT TERMS (key words)			15. NUMBER OF PAGES
model-based recognition	image understanding	vision	125
object recognition	region growing	grouping	16. PRICE CODE
image	recognition		

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED

Block 13 continued:

that satisfy the constraints imposed by the model's shape. But, in addition to these constraints, there are important *match-independent* constraints derived from general information about the world, the imaging process, and the library of models as a whole. These constraints are less strict than match-dependent shape constraints, but they can be efficiently applied without the combinatorics of matching. In this thesis, I present two specific modules that demonstrate the utility of match-independent constraints. The first is a *region-based grouping* mechanism that drastically reduces the combinatorics of choosing subsets of features. Instead of all subsets, it finds groups of image features that are likely to come from a single object (without hypothesizing which object). Then, in order to address the combinatorics of matching within each subset, the second module, *interpretive matching*, makes explicit hypotheses about occlusion and instabilities in the image features. This module also begins to make matches with the model features, and applies only those match-dependent constraints that are independent of the model pose. Together, the two modules sharply reduce the number of matches that need be subjected to pose-dependent model shape constraints. The modules are demonstrated in the context of a complete recognition system, Reggie.

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**